TUGAS AKHIR – KI1502

KONTROL GERAK ROBOT DENGAN SINYAL *ELECTROENCEPHALOGRAM* (EEG) EMOTIV EPOC NEUROHEADSET

SANDY AKBAR DEWANGGA
NRP 5111100092

Dosen Pembimbing I
Prof.Ir. Handayani Tjandrasa, M.Sc, Ph.D

Dosen Pembimbing II
Dr. Darlis Herumurti, S.Kom., M.Kom.

JURUSAN TEKNIK INFORMATIKA
Fakultas Teknologi Informasi
Institut Teknologi Sepuluh Nopember
Surabaya 2015

[Halaman ini sengaja dikosongkan]

UNDERGRADUATE THESES – KI1502

ROBOT MOTION CONTROL WITH ELECTROENCEPHALOGRAM SIGNALS (EEG) EMOTIV EPOC NEUROHEADSET

SANDY AKBAR DEWANGGA
NRP 5111100092

Supervisor I
Prof.Ir. Handayani Tjandrasa, M.Sc, Ph.D

Supervisor II
Dr. Darlis Herumurti, S.Kom., M.Kom.

DEPARTMENT OF INFORMATICS
FACULTY OF INFORMATION TECHNOLOGY
INSTITUT TEKNOLOGI SEPULUH NOPEMBER
SURABAYA 2015

[Halaman ini sengaja dikosongkan]

LEMBAR PENGESAHAN

KONTROL GERAK ROBOT DENGAN SINYAL *ELECTROENCEPHALOGRAM* (EEG) EMOTIV EPOC NEUROHEADSET

TUGAS AKHIR

Diajukan Untuk Memenuhi Salah Satu Syarat
Memperoleh Gelar Sarjana Komputer
pada
Bidang Studi Komputasi Cerdas dan Visi
Program Studi S-1 Jurusan Teknik Informatika
Fakultas Teknologi Informasi
Institut Teknologi Sepuluh Nopember

Oleh
SANDY AKBAR DEWANGGA
NRP : 5111 100 092

Disetujui oleh Dosen Pembimbing Tugas Akhir:

1. Prof.Ir Handayani Tjandrasa, M.Sc, Ph.D ..... (Pembimbing 1)
   NIP: 19490823 197603 2 001

2. Dr. Darlis Herumurti, S.Kom., M.Kom. ..... (Pembimbing 2)
   NIP: 19771217 200312 1 001

SURABAYA
JUNI, 2015

[Halaman ini sengaja dikosongkan]

KONTROL GERAK ROBOT DENGAN SINYAL *ELECTROENCEPHALOGRAM* (EEG) EMOTIV EPOC NEUROHEADSET

Nama Mahasiswa	**: SANDY AKBAR DEWANGGA**
NRP	**: 5111100092**
Jurusan	**: Teknik Informatika FTIF-ITS**
Dosen Pembimbing 1	**: Prof.Ir Handayani Tjandrasa, M.Sc, Ph.D**
Dosen Pembimbing 2	**: Dr. Darlis Herumurti, S.Kom., M.Kom.**

Abstrak

Ide untuk menggerakkan robot atau perangkat dengan hanya berpikir (aktivitas otak dengan subjek manusia) menjadi topik yang menarik bagi para peneliti beberapa tahun terakhir seiring dengan semakin berkembangnya pemahaman tentang otak manusia. Dengan menangkap transmisi sinyal langsung dari otak manusia atau electroencephalogram (EEG), kita dapat menjadikan pikiran seseorang sebagai perintah gerak untuk robot.

Tugas akhir ini menggabungkan metode Linear Discriminant Analysis (LDA) dan Support Vector Machine (SVM) untuk pembentukan model terbaik. Model terbaik diperoleh dari proses training fitur dataset sinyal EEG yang didapatkan dari subjek secara non-invasif dengan tingkat kesalahan paling kecil. Model terbaik kemudian digunakan untuk mendapatkan perintah gerak robot secara realtime.

Dari uji coba, didapatkan tingkat akurasi rata-rata 69.90% dan didapatkan akurasi terbaik sebesar 73.03%, sehingga algoritma ini dapat digunakan dalam pembentukan model klasifikasi perintah gerak robot. Tugas akhir ini dapat dijadikan salah satu alternatif pembentukan model terbaik

klasifikasi perintah gerak robot dengan pikiran secara realtime.

Kata kunci: Emotiv EPOC, Electroencephalogram, Linear Discriminant Analisis, Support Vector Machine, Kontrol Robot, Non-invasif, Realtime

ROBOT MOTION CONTROL WITH ELECTROENCEPHALOGRAM SIGNALS (EEG) EMOTIV EPOC NEUROHEADSET

Student's Name	**: SANDY AKBAR DEWANGGA**
Student's ID	**: 5111100092**
Department	**: Teknik Informatika FTIF-ITS**
First Advisor	**: Prof.Ir Handayani Tjandrasa, M.Sc, Ph.D**
Second Advisor	**: Dr. Darlis Herumurti, S.Kom., M.Kom.**

Abstract

The idea to move the robot or device by simply thinking (brain activity with human subjects) become a topic of interest for researchers in recent years due to the growing understanding of the human brain. By capturing the transmission of signals directly from the human brain or electroencephalogram (EEG), we can make a person's mind as motion commands to the robot.

This final project combines the methods of Linear Discriminant Analysis (LDA) and Support Vector Machine (SVM) for the establishment of the best model. The best model obtained from the training dataset features EEG signals obtained from the subject non-invasively with minimum error. The best model is then used to get the robot motion commands in realtime.

From experiments, obtained an average accuracy rate of 69.90% and obtained the best accuracy of 73.03%, so the algorithm can be used classification model robot motion commands. This thesis can be used as one alternative to the establishment of the best models in the classification of robot motion commands with the mind in realtime.

Keywords: Emotiv EPOC, Electroencephalogram, Linear Discriminant Analysis, Support Vector Machine, Robot Control, Non-invasive, Realtime

KATA PENGANTAR

Segala puji bagi Allah SWT yang telah melimpahkan rahmat dan hidayah-Nya sehingga penulis dapat menyelesaikan Tugas Akhir yang berjudul **"KONTROL GERAK ROBOT DENGAN SINYAL *ELECTROENCEPHALOGRAM* (EEG) EMOTIV EPOC NEUROHEADSET".**

Pengerjaan Tugas Akhir ini merupakan suatu kesempatan yang sangat baik bagi penulis. Dengan pengerjaan Tugas Akhir ini, penulis bisa belajar lebih banyak untuk memperdalam dan meningkatkan apa yang telah didapatkan penulis selama menempuh perkuliahan di Teknik Informatika ITS. Dengan Tugas Akhir ini penulis juga dapat menghasilkan suatu implementasi dari apa yang telah penulis pelajari.

Selesainya Tugas Akhir ini tidak lepas dari bantuan dan dukungan beberapa pihak. Sehingga pada kesempatan ini penulis mengucapkan syukur dan terima kasih kepada:

1. Allah SWT dan Nabi Muhammad SAW.
2. Papa, Mama, Bima, dan Danar yang telah memberikan dukungan moral dan material serta do'a yang tak terhingga untuk penulis. Serta selalu memberikan semangat dan motivasi pada penulis dalam mengerjakan Tugas Akhir ini.
3. Ibu Prof.Ir Handayani Tjandrasa, M.Sc, Ph.D selaku pembimbing I yang telah membantu, membimbing, dan memotivasi penulis dalam menyelesaikan Tugas Akhir ini dengan sabar.
4. Bapak Dr. Darlis Herumurti, S.Kom., M.Kom. selaku pembimbing II yang juga telah membantu, membimbing, dan memotivasi kepada penulis dalam mengerjakan Tugas Akhir ini.
5. Ibu Dr. Eng. Nanik Suciati, S.Kom., M.Kom. selaku Kepala Jurusan Teknik Informatika ITS, Bapak Radityo Anggoro, S.Kom.,M.Sc. selaku koordinator TA, dan segenap dosen Teknik Informatika yang telah memberikan ilmunya.

6. Rahmatullah Hutami Hidayat yang selalu menjadi teman yang setia dan tanpa lelah memberikan motivasi serta bantuan kepada penulis dalam mengerjakan tugas akhir.
7. Para sahabat sepenanggung-seperjuangan: Siful, Ajeng, Alfin, dan Aank yang senantiasa mengisi hari-hari penulis baik suka maupun duka.
8. Teman-teman Admin Microsoft Mobility yang selalu menjadi teman diskusi dan berbagi ilmu.
9. Teman-teman Pengurus Harian HMTC 2013-2014 Kabinet Bersahabat dan teman-teman angkatan 2011 yang yang telah membantu, berbagi ilmu, menjaga kebersamaan, dan memberi motivasi kepada penulis, kakak-kakak angkatan 2009 dan 2010 serta adik-adik angkatan 2012 dan 2013 yang membuat penulis untuk selalu belajar.
10. Serta semua pihak yang yang telah turut membantu penulis dalam menyelesaikan Tugas Akhir ini.

Penulis menyadari bahwa Tugas Akhir ini masih memiliki banyak kekurangan. Sehingga dengan kerendahan hati, penulis mengharapkan kritik dan saran dari pembaca untuk perbaikan ke depannya.

Surabaya, Juni 2015

DAFTAR ISI

[Halaman ini sengaja dikosongkan]

DAFTAR TABEL

[Halaman ini sengaja dikosongkan]

DAFTAR GAMBAR

BAB I
PENDAHULUAN

1.1 Latar Belakang

Ide untuk menggerakkan robot atau perangkat tidak dengan kontrol manual melainkan dengan hanya berpikir (aktivitas otak dengan subjek manusia) menjadi topik yang menarik bagi para peneliti beberapa tahun terakhir. Berkembangnya teknologi komputer modern berjalan seiring dengan pemahaman tentang otak manusia [1]. Otak manusia dipenuhi dengan neuron, sel-sel saraf individu yang terhubung satu sama lain dengan dendrit dan akson. Setiap kali manusia berpikir, bergerak, merasakan atau mengingat sesuatu, neuron akan berkerja dan membangkitkan sinyal-sinyal listrik kecil yang merambat dari neuron ke neuron dengan kecepatan yang mencapai 250 mph [2].

Ketika manusia bergerak atau berpikir untuk bergerak, maka neuron pada motor *cortex-area* pada cerebrum otak yang memerintahkan gerak akan memproduksi aliran sinyal listrik kecil. Dengan menangkap transmisi sinyal langsung dari otak manusia atau *electroencephalogram* (EEG), kita dapat menjadikan pikiran seseorang sebagai perintah gerak untuk robot [3].

Saat ini terdapat berbagai perangkat untuk menangkap sinyal EEG dengan kualitas yang baik dan harga yang terjangkau. Emotiv EPOC Neuroheadset adalah salah satu diantaranya. Emotiv EPOC Neuroheadset memiliki 14 elektroda saline dan 2 elektroda referensi yang digunakan untuk mengumpulkan data EEG dari pengguna, dan mengirimkan informasi secara nirkabel ke komputer .

Klasifikasi perintah gerak robot dilakukan dengan kombinasi antara metode *Linear Discriminant Analysis* dan *Support Vector Machine* sehingga didapatkan model terbaik. Model terbaik diperoleh dari hasil training fitur dataset sinyal EEG perintah pikiran dengan tingkat kesalahan yang paling kecil. Terdapat 5 kelas perintah gerak yang digunakan yaitu maju,

mundur, belok kanan, belok kiri, dan netral. Model terbaik kemudian digunakan untuk mendapatkan kelas perintah gerak untuk menggerakkan robot secara *realtime*.

1.2 Rumusan Masalah

Rumusan masalah yang diangkat dalam Tugas Akhir ini dapat dipaparkan sebagai berikut:

1. Bagaimana proses mendapatkan sinyal EEG serta spesifikasi sinyal yang dihasilkan dari perangkat Emotiv EPOC Neuroheadset?
2. Bagaimana proses pendekodean sinyal EEG menjadi perintah gerak robot?
3. Bagaimana proses pengiriman perintah gerak kepada robot?

1.3 Batasan Masalah

Permasalahan yang dibahas dalam Tugas Akhir ini memiliki beberapa batasan, yaitu sebagai berikut:

1. Data yang diambil adalah data sinyal EEG subjek uji dan berada pada kondisi tenang serta tidak terdapat banyak gerakan yang mengganggu.
2. Implementasi program dilakukan pada lingkungan komputer desktop.
3. Robot yang digunakan adalah robot beroda yg terhubung secara nirkabel dengan komputer desktop serta dapat bergerak maju, mundur, belok kiri, dan belok kanan.
4. Subjek uji berjumlah 1 orang.

1.4 Tujuan

Tujuan dari tugas akhir ini adalah untuk membangun sistem perintah gerak robot dari sinyal EEG yang dibangkitkan oleh perangkat Emotiv EPOC Neuroheadset.

1.5 Manfaat

Tugas akhir ini diharapkan mampu membangun sebuah sistem yang dapat memerintahkan gerak robot hanya dengan aktivitas pikiran manusia.

1.6 Metodologi

Tahapan-tahapan yang dilakukan dalam pengerjaan Tugas Akhir ini adalah sebagai berikut:

1. Penyusunan proposal Tugas Akhir.
   Tahap awal untuk memulai pengerjaan Tugas Akhir adalah penyusunan proposal Tugas Akhir. Proposal Tugas Akhir yang diajukan memiliki gagasan yang sama dengan Tugas Akhir ini, yaitu implementasi metode *Linier Discriminant Analysis* dan *Support Machine Vector* untuk melakukan klasifikasi data.

2. Studi literatur
   Literatur yang dipelajari pada pengerjaan Tugas Akhir ini berasal jurnal ilmiah yang diambil dari IEEE. Makalah yang digunakan sebagai acuan adalah "*Noninvasive brain-actuated control of a mobile robot by human EEG*", "*Electroencephalogram-Based Control of an Electric Wheelchair*" dan "*Electroencephalogram-based control of a mobile robot*" serta beberapa referensi lain yang berhubungan dengan topik yang diajukan.

3. Analisis dan perancangan

   Pada tahap ini dilakukan analisis dan desain perancangan aplikasi sesuai dengan tujuan yang dijabarkan serta perangkat robot yang digunakan. Selain itu, pada tahap ini dilakukan pengujian perangkat Emotiv EPOC Neuroheadset, penyesuaian dengan ruang lingkup impelementasi aplikasi dan pengambilan data mentah dari subjek uji.

4. Implementasi

   Pada tahap ini dilaksanakan implementasi metode dan algoritma yang telah direncanakan. Implementasi perangkat lunak dilakukan di dalam platform desktop.

5. Pengujian dan evaluasi

   Pada tahap ini dilakukan uji coba dengan menggunakan perangkat Emotiv EPOC Neuroheadset. Subjek uji diharuskan untuk melakukan perintah gerak *high-level* langsung dalam pikiran dengan membayangkan atau memikirkan suatu perintah. Gelombang otak yang dihasilkan ditangkap kemudian diproses untuk menghasilkan perintah gerak pada robot. Akurasi diukur dengan menghitung banyak ketepatan hasil deteksi sistem dengan *groundtruth*. *Power* diukur dengan menghitung banyak sekuen ketepatan hasil deteksi sistem dengan *groundtruth.*

6. Penyusunan Buku Tugas Akhir.

   Pada tahapan ini disusun buku yang memuat dokumentasi mengenai pembuatan serta hasil dari implementasi perangkat lunak yang telah dibuat.

1.7 Sistematika Penulisan Laporan Tugas Akhir

Buku Tugas Akhir ini bertujuan untuk mendapatkan gambaran dari pengerjaan Tugas Akhir ini. Selain itu, diharapkan

dapat berguna untuk pembaca yang tertarik untuk melakukan pengembangan lebih lanjut. Secara garis besar, buku Tugas Akhir terdiri atas beberapa bagian seperti berikut ini:

Bab I Pendahuluan

Bab yang berisi mengenai latar belakang, tujuan, dan manfaat dari pembuatan Tugas Akhir. Selain itu bab ini juga berisi permasalahan, batasan masalah, metodologi yang digunakan, dan sistematika penulisan.

Bab II Tinjauan Pustaka

Bab ini berisi penjelasan secara detail mengenai dasar-dasar penunjang dan teori-teori yang digunakan untuk mendukung pembuatan Tugas Akhir ini.

Bab III Perancangan Sistem

Bab ini berisi tentang rancangan keseluruhan sistem. Penjelasan berupa *pseudocode* fungsi – fungsi utama pada program.

Bab IV Implementasi

Bab ini membahas implementasi dari rancangan yang telah dibuat pada bab sebelumnya. Penjelasan berupa kode program yang digunakan untuk proses implementasi.

Bab V Uji Coba Dan Evaluasi

Bab ini menjelaskan kemampuan perangkat lunak dengan melakukan pengujian kebenaran dan pengujian kinerja dari sistem yang telah dibuat.

Bab VI Kesimpulan Dan Saran

Bab ini berisi kesimpulan dari hasil uji coba yang dilakukan dan saran untuk pengembangan perangkat lunak ke depannya.

[Halaman ini sengaja dikosongkan]

BAB II
TINJAUAN PUSTAKA

Bab ini berisi penjelasan teori-teori yang berkaitan dengan algoritma yang diajukan pada pengimplementasian program. Penjelasan ini bertujuan untuk memberikan gambaran secara umum terhadap program yang dibuat dan berguna sebagai penunjang dalam pengembangan perangkat lunak.

2.1 Sinyal EEG (*Electroencephalogram*)

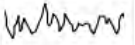
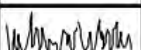
Sinyal EEG adalah sinyal aktivitas listrik dilapisan terluar kulit otak (*cerebral cortex*). Umumnya sinyal EEG digunakan untuk mendiagnosa penyakit yang berkaitan dengan otak dan kejiwaan, seperti epilepsi, tumor otak, mendeteksi posisi/lokasi otak yang terluka, serta membantu diagnosa gangguan mental. Pengukuran sinyal EEG dilakukan dengan cara meletakkan elektroda-elektroda pada kulit kepala (*scalp*) dan hasil pengukurannya dipengaruhi oleh beberapa variabel, seperti kondisi mental, aktivitas pada saat pengukuran, kondisi kesehatan, dan kondisi lingkungan pengukuran, serta faktor stimulus eksternal yang diterima pada saat pengukuran. Karena beberapa karakteristik dari sinyal EEG yang tidak periodik, tidak mempunyai pola baku, serta mempunyai amplitudo tegangan yang kecil, menyebabkan sinyal EEG sangat mudah tertimbun *noise* sehingga sulit untuk didiagnosa kecuali oleh dokter/ahli terlatih.

Perekaman sinyal EEG, selain dipengaruhi oleh kondisi mental, aktifitas pasien, dan kondisi lingkungan disekitar pengukuran, juga dipengaruhi oleh pemberian stimulus dari eksternal saat pengukuran, seperti suasana relaks namun sadar, suasana berpikir/beraktifitas dengan mata terbuka, suasana tidur ringan dengan memberi tekanan berupa cahaya lampu yang dihidup matikan, serta suasana tidur nyenyak.

Untuk menganalisa adanya penyakit di otak, maka dilakukan pengukuran sinyal EEG berdasarkan kondisi pikiran, frekuensi,

dan amplitudo tegangan. Berdasarkan frekuensi, amplitudo tegangan, dan kondisi obyek, sinyal EEG dapat dibagi menjadi empat macam gelombang, yaitu gelombang alpha, beta, theta, dan delta. Pembagian macam gelombang EEG diperlihatkan pada Tabel 2.1.

Tabel 2.1. Jenis gelombang otak.

Jenis Gel.	Bentuk Gel	Frekwensi	Tegangan (V)	Kondisi Obyek
Alpha		8 - 13 Hz	anak:75 μ dewasa:50 μ	Relaks, Mata-tertutup
Betha		> 14 Hz	10 - 20 μ	aktifitas/ berpikir, mata terbuka
Theta		4 - 7 Hz	anak: 50μ dewasa:10μ	tidur ringan/ stres,emosional
Delta		0.5 - 3 Hz	1 m	tidur nyenyak

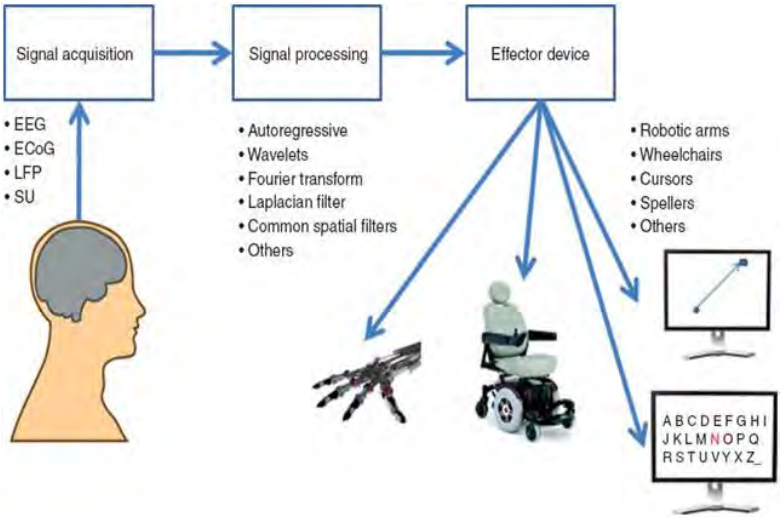
2.2 *Brain-Computer Interface* (BCI)

Brain-Computer Interface (BCI), disebut juga *Mind-Machine Interface* (MMI), *Direct Neural Interface* (DNI), *Synthetic Telepathy Interface* (STI) atau *Brain-Machine Interface* (BMI) adalah jalur komunikasi langsung antara otak dan perangkat eksternal. BCI sering diarahkan untuk membantu menambah atau memperbaiki fungsi kognitif atau sensorik-motorik manusia [7]. BCI dapat dibagi menjadi dua kategori: invasif dan non-invasif [8].

BCI invasif adalah alat yang dipasang langsung pada materi abu-abu otak. Alat tersebut dapat menerima sinyal langsung dari otak dengan paling jelas dan efektif. BCI invasif telah digunakan secara efektif pada hewan percobaan dan manusia untuk berbagai aplikasi, termasuk pengobatan kebutaan bawaan. Kelemahan dari

BCI invasif adalah biaya operasi yang mahal serta berbahaya karena seringkali tubuh meresponnya sebagai benda asing. Sedangkan pada BCI non-invasif, tidak perlu dilakukan penanaman alat pada otak. Alat cukup ditempelkan pada kulit permukaan kepala sehingga BCI non-invasif lebih murah dan aman dibandingkan dengan BCI invasif [9].

Cara kerja BCI dan kaitannya dengan otak manusia ditunjukkan pada Gambar 2.1. [10].



Gambar 2.1. Skema kerja BCI dan otak manusia.

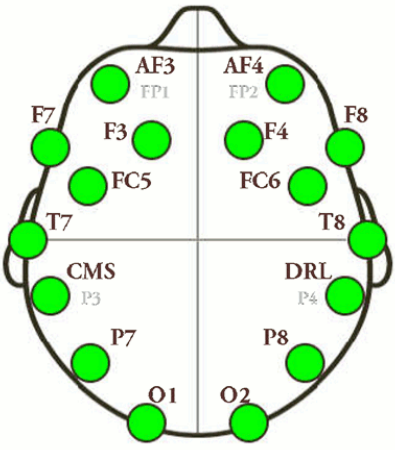
2.3 Emotiv EPOC Neuroheadset

Emotiv EPOC Neuroheadset adalah alat pemindai aktivitas otak yang menggunakan empat belas sensor dan dua titik referensi untuk melacak empat belas saluran yang berbeda dari aktivitas elektroensefalografi. Emotiv EPOC mengumpulkan data dari saluran standar: AF3, AF4, F3, F4, F7, F8, FC5, FC6, P3 (CMS),

P4 (DRL), P7, P8, T7, T8, O1, dan O2. Penampang Emotiv EPOC Neuroheadset dapat dilihat pada Gambar 2.2.

Gambar 2.2. Emotiv EPOC Neuroheadset.

Emotiv EPOC menggunakan metode sampling sekuensial dan beroperasi pada resolusi 14 bit atau 16 bit per kanal dengan respon frekuensi antara 0.16-43 Hz. Lokasi sensor Emotiv EPOC ditunjukkan pada Gambar 2.3.



Gambar 2.3. Lokasi sensor Emotiv EPOC.

Quantitative Electroencephalography (QEEG) (Dr. Horst H, 2013) adalah teknik pencitraan otak yang memungkinkan kita untuk memahami aktivitas otak dan setiap bagiannya [11]. Penelitian ini memberikan kita informasi yang berguna untuk memahami arti dari setiap sensor Emotiv EPOC. Pada Tabel 2.2 diperlihatkan daftar kanal sensor Emotiv EPOC beserta fungsinya pada otak.

Tabel 2.2. Fungsional otak di setiap sensor Emotiv EPOC.

Channel	Brain Function
AF3	Attention
AF4	Judgment
F3	Motor planning
F4	Motor planning for left upper
F7	Verbal Expression
F8	Emotional Expression: Anger, Happy
FC5	Right Body controller
FC6	Left Body controller
T7	Verbal memory
T8	Emotional memory
P7	Verbal understanding
P8	Emotional: Understanding, Motivation
O1	Visual processing
O2	Visual processing

2.4 *Linier Discriminant Analysis* (LDA)

LDA adalah salah satu teknik statistik yang bisa digunakan pada hubungan dependensi (hubungan antar variabel dimana sudah bisa dibedakan mana variabel respon dan mana variabel penjelas). Lebih spesifik lagi, analisis diskriminan digunakan pada kasus

dimana variabel respon berupa data kualitatif dan variabel penjelas berupa data kuantitatif. Analisis diskriminan bertujuan untuk mengklasifikasikan suatu individu atau observasi ke dalam kelompok yang saling bebas (*mutually exclusive*/*disjoint*) dan menyeluruh (*exhaustive*) berdasarkan sejumlah variabel penjelas.

Ada dua asumsi utama yang harus dipenuhi pada analisis diskriminan ini, yaitu:

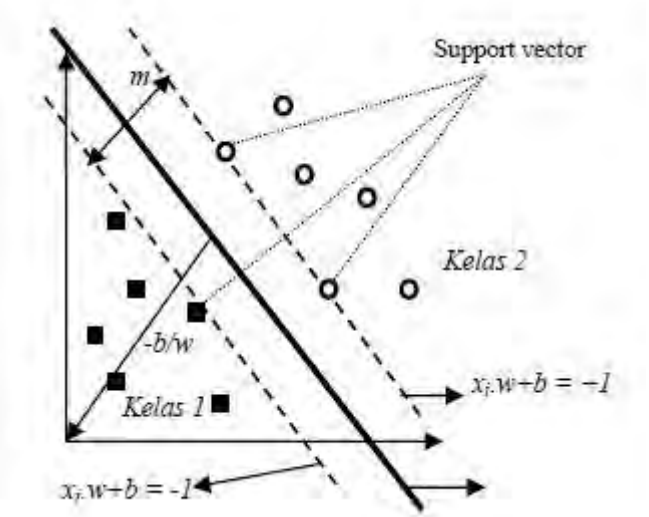
1. Sejumlah p variabel penjelas harus berdistribusi normal.
2. *Matriks varians-covarians* variabel penjelas berukuran p *x* p pada kedua kelompok harus sama.

Jika dianalogikan dengan regresi *linier*, maka analisis diskriminan merupakan kebalikannya. Pada regresi linier, variabel respon yang harus mengikuti distribusi normal dan homoskedastis, sedangkan variabel penjelas diasumsikan *fixed*, artinya variabel penjelas tidak disyaratkan mengikuti sebaran tertentu. Untuk analisis diskriminan, variabel penjelasnya seperti sudah disebutkan di atas harus mengikuti distribusi normal dan homoskedastis, sedangkan variabel responnya *fixed*.

2.5 *Support Vector Machine* (SVM)

SVM berusaha menemukan *hyperplane* yang terbaik pada *input space*. Prinsip dasar SVM adalah *linear classifier* yang selanjutnya dikembangkan agar dapat bekerja pada permasalahan non-linear dengan memasukkan konsep *kernel trick* pada ruang kerja berdimensi tinggi [4]. SVM dapat melakukan klasifikasi data yang terpisah secara linier (*linearly separable*) dan *non*-linier (*nonlinear separable*) [5]. *Linearly separable* data merupakan data yang dapat dipisahkan secara linier. Misalkan $\{x_1,..., x_n\}$ adalah *dataset* dan $x_i \in R^d$, serta $y_i \in \{+1,-1\}$ adalah label kelas dari data x_i. Anggap ada beberapa *hyperplane* yang memisahkan sampel positif dan negatif, maka *x* yang berada pada *hyperplane* memenuhi persamaan $w.x + b = 0$. Untuk permasalahan data linier, algoritma *support vector* hanya mencari *hyperplane* dengan margin yang terbesar (jarak antara dua kelas pola). *Hard margin hyperplane*

ditunjukkan pada Gambar 2.4. *Hyperplane* terbaik tidak hanya dapat memisahkan data dengan baik tetapi juga yang memiliki margin paling besar. Data yang berada pada bidang pembatas ini diseb



Gambar 2.4. Hard Margin Hyperplane.

Untuk menyelesaikan permasalahan data *non*-linier dalam SVM adalah dengan cara memetakan data ke ruang dimensi lebih tinggi (ruang fitur atau *feature space*) [5], dimana data pada ruang tersebut dapat dipisahkan secara linier, dengan menggunakan transformasi Φ.

$$\Phi : \Re^d \mapsto H \qquad (2.1)$$

SVM pertama kali dikembangkan oleh Vapniks untuk klasifikasi biner, namun selanjutnya dikembangkan untuk klasifikasi *multiclass* (banyak kelas). Pendekatannya adalah dengan membangun *multiclass classifier*, yaitu dengan cara

menggabungkan beberapa SVM *biner*. Pendekatan ini terdiri dari metode satu lawan semua (*One Against All*) dan metode satu lawan satu (*One Against One*) [6].

2.6 Normalisasi

Normalisasi adalah teknik *preprocessing* untuk melakukan penskalaan dan penstandaran data. Dua contoh normalisasi yang dikenal adalah normalisasi skala dan normalisasi *Gaussian* (standardisasi).

2.6.1 Normalisasi Skala

Normalisasi skala yaitu melakukan penskalaan data pada rentang tertentu, dimana rentang yang umum digunakan adalah rentang 0-1. Normalisasi ini bertujuan untuk menghindari rentang data yang terlalu jauh. Rumus untuk menghitung normalisasi skala adalah:

$$Y = \frac{X - X_{min}}{X_{max} - X_{min}} \tag{2.2}$$

2.6.2 Normalisasi *Gaussian*

Standardisasi adalah proses pembuatan data berdistribusi normal (0, 1). Proses ini sangat berguna untuk mengecilkan varians secara total, sehingga tidak ada dominasi varians pada beberapa data/atribut. Rumus untuk menghitung standardisasi adalah sebagai berikut:

$$Y = \frac{X - mean}{standar\ devasi} \tag{2.3}$$

2.7 Filter Butterworth

Filter butterworth adalah jenis filter pengolahan sinyal yang dirancang untuk memiliki sebuah *flat* respon frekuensi yang memungkinkan dalam *passband* sehingga disebut juga *maximally*

flat magnitude filter. Filter butterworth *lowpass* orde ke-N dengan frekuensi *cutoff* adalah:

$$|H_{Butterwot\ h}(\omega)|^2 = \frac{1}{1+\omega^{2N}} \quad (2.4)$$

Respon frekuensi dari filter butterworth adalah maksimal datar (tidak memiliki riak) di passband dan memiliki pelemahan yang cukup tajam pada frekuensi *stopband*. Filter butterworth memberikan optimasi pada daerah *passband*.

[Halaman ini sengaja dikosongkan]

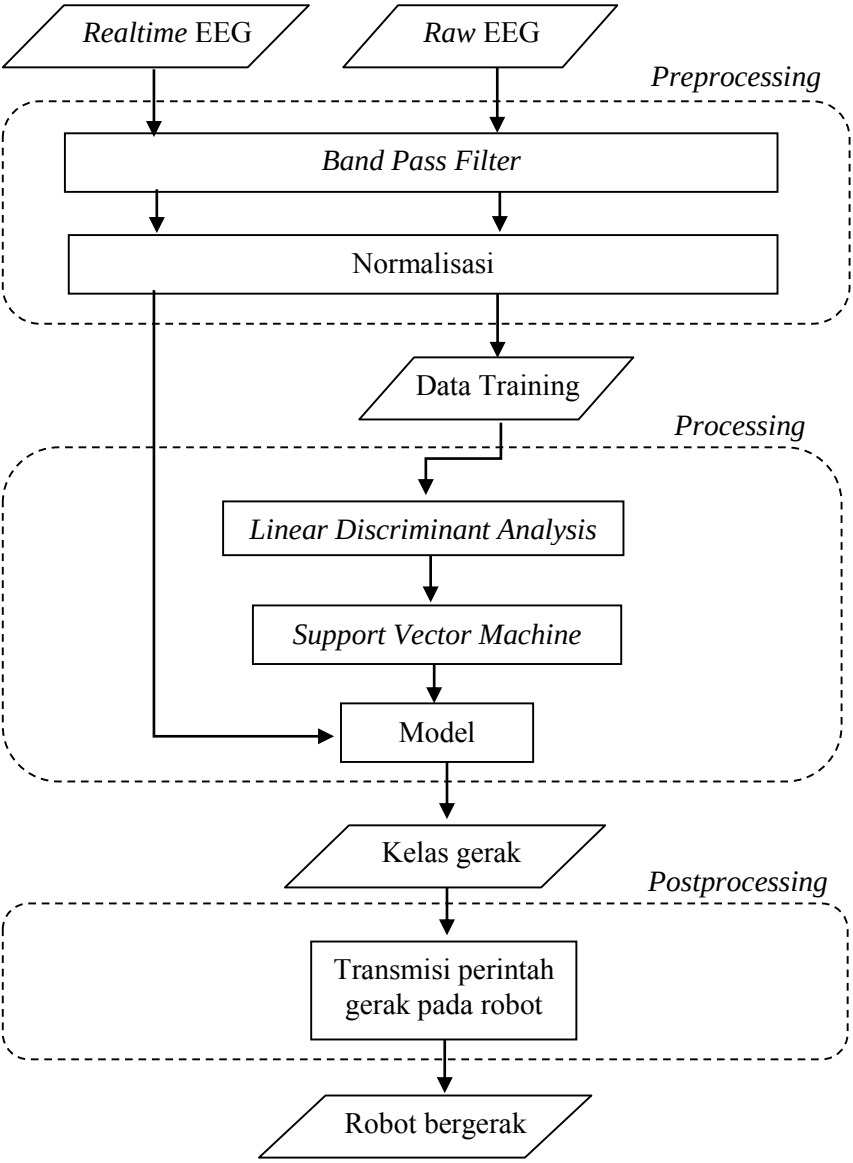
BAB III
PERANCANGAN SISTEM

Pada bab ini dijelaskan mengenai perancangan sistem yang dibuat. Dalam pembuatan suatu sistem, perancangan secara teknis tentang bagaimana aplikasi berjalan merupakan bagian awal yang amat penting dan harus dicermati. Sehingga bab ini secara khusus dibuat untuk menjelaskan perancangan sistem yang dibuat dalam tugas akhir ini, mulai dari desain metode secara umum, perancangan data, *preprocessing, processing, postprocessing,* sampai perancangan kode program.

3.1 Desain Metode Secara Umum

Pada tugas akhir ini dibangun suatu sistem yang dapat menggerakkan robot dengan menggunakan sinyal EEG yang dibangkitkan oleh perangkat Emotiv EPOC Neuroheadset. Untuk dapat menggerakkan robot, maka digunakan proses klasifikasi untuk menerjemahkan data *realtime* EEG subjek uji sehingga didapatkan kelas perintah gerak yang kemudian dikirimkan ke robot. Proses klasifikasi mengguakan metode gabungan antara LDA dan SVM. Untuk dapat melakukan proses klasifikasi data EEG diperlukan perekaman data EEG terlebih dahulu untuk mendapatkan data *raw* EEG untuk dijadikan sebagai data *training.*

Data *raw* EEG diperoleh dari hasil rekaman EEG subjek uji dengan menggunakan Emotiv EPOC. Data *raw* EEG kemudian digunakan sebagai data training setelah sebelumnya dilakukan *filter* dan normalisasi terlebih dahulu sehingga didapatkan model. Data *realtime* EEG kemudian diklasifikasi dengan memakai model yang telah dibentuk setelah sebelumnya dilakukan juga filter dan normalisasi sehingga didapatkan kelas gerak. Kelas gerak kemudian dikirimkan pada robot. Alur dari seluruh proses ditunjukkan oleh Gambar 3.1.



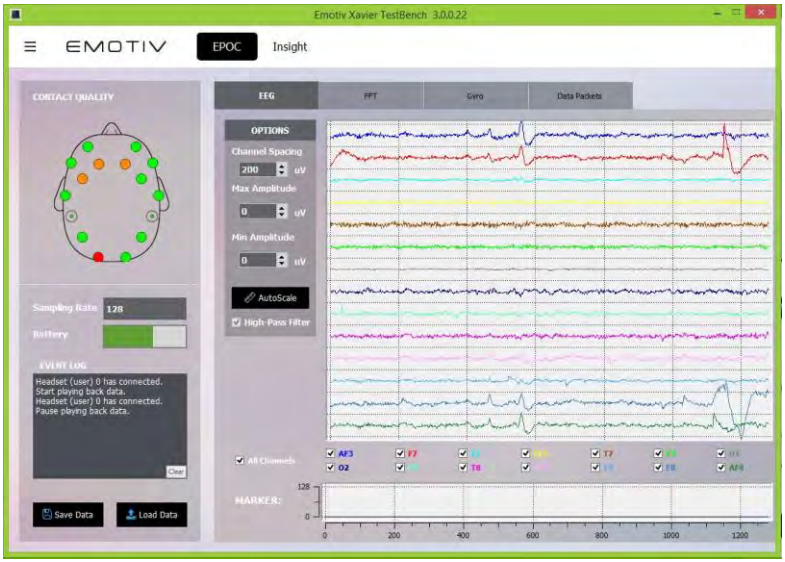
Gambar 3.1. Alur proses program

3.2 Perancangan Data

Proses pengambilan data EEG ada dua bagian, pertama adalah perekaman data *raw* EEG sedangkan yang kedua adalah pengambilan data *realtime* EEG.

3.2.1 *Raw* Data EEG

Raw data EEG didapatkan dari hasil perekaman data EEG subjek uji dengan menggunakan Emotiv EPOC. Proses perekaman data EEG dilakukan dengan menggunakan perangkat lunak Emotiv Xavier TestBench. Pada Gambar 3.2 tampak tampilan antarmuka Emotiv Xavier TestBench.



Gambar 3.2. Tampilan antarmuka Emotiv Xavier TestBench

3.2.2 *Realtime* Data EEG

Data EEG pada Emotiv EPOC dapat diakses secara *realtime* dengan menggunakan SDK library dari Emotiv.

3.3 *Preprocessing*

Dalam mengawali proses pengerjaan program, maka harus disiapkan terlebih dahulu data yang akan digunakan dalam proses pengerjaan. *Preprocessing* adalah tahapan yang penting untuk dilakukan. Pada tugas akhir kali ini, tahap *preprocessing* yang dilakukan adalah *filter noise* dan normalisasi.

3.3.1 *Filter Noise*

Gelombang EEG sangat rentan sekali dengan *noise* karena sinyal listrik yang berusaha ditangkap berukuran kecil. Untuk itu diperlukan *filter* agar data yang dihasilkan memiliki kualitas yang bagus. *Filter noise* yang digunakan adalah Butterworth Band Pass Filter order 4 dengan nilai minimal = 29 dan maksimal = 40.

3.3.2 Normalisasi

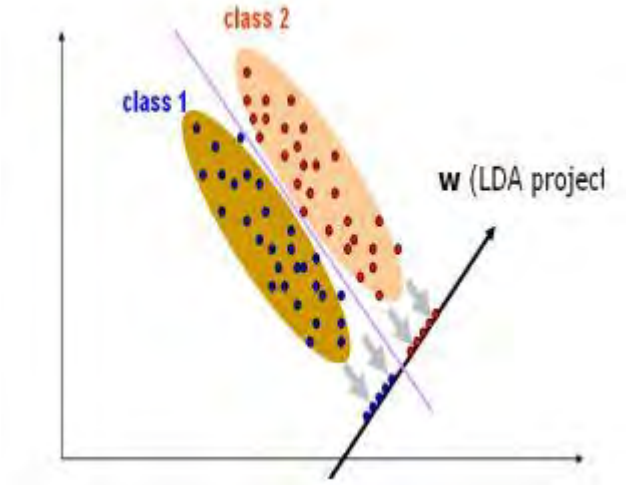
Data sampel dari Emotiv EPOC memiliki 14 atribut dari setiap kanal sensor dengan 128 *record* untuk setiap detiknya. Data ini kemudian dinormalisasi secara terspisah untk setiap kanal. Normalisasi dilakukan setiap 0.5s data (64 *record*). Rumus normalisasi yang digunakan adalah sebagai berikut.

$$Y = \frac{X - \bar{X}}{30} \qquad (3.1)$$

Dimana nilai 30 adalah rata-rata amplitudo sinyal yang dihasilkan.

3.4 *Processing*

Untuk proses data training, digunakan metode gabungan antara LDA dan SVM. Ide dasar dari LDA adalah menemukan sebuah transformasi linear sehingga pengklasteran dapat dipisahkan setelah transformasi. Ini dapat diperoleh melalui analisa matriks *scatter*. LDA mengelompokkan vektor data dari kelas yang sama dan memisahkan kelas yang berbeda. Visualisasi proyeksi data 2 kelas dengan menggunakan LDA dapat dilihat pada gambar 3.3.



Gambar 3.3. Proyeksi data 2 kelas LDA

LDA dilakukan untuk mendapatkan sebaran data dengan tingkat diskriminan yang optimal. Hasil proyeksi dari LDA digunakan sebagai data masukan dari SVM. Model SVM kemudian berusaha memprediksi di dalam kelas mana titik-titik data tersebut termasuk.

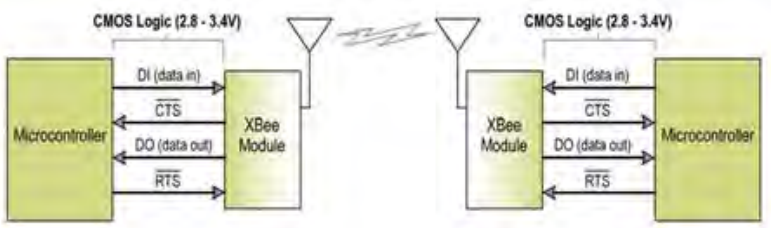
3.5 *Postprocessing*

Setelah kelas gerak hasil klasifikasi didapatkan, perintah gerak perlu dikirimkan kepada robot agar robot dapat bergerak. Rancangan untuk robot yang digunakan adalah robot beroda yang dapat bergerak maju, mundur, belok kanan, dan belok kiri. Robot terhubung dengan program BCI secara nirkabel dengan menggunakan *transmitter* XBee Pro.

XBee Pro adalah RF Module (Radio Frequency) yang beroperasi pada frekuensi 2.4 GHz. Xbee Pro merupakan teknologi Zigbee yang berada dalam golongan WPAN (Wireless Personal Area Network) yang memiliki kecepatan transfer sebesar 250kbps. Untuk menggunakan modul Xbee ini minimal membutuhkan 2 buah modul untuk melakukan komunikasi point to point atau lebih untuk komunikasi multipoint. Bentuk fisik dari XBee PRO ditunjukkan pada Gambar 3.4.

XBee PRO beroperasi pada tegangan 2.8V – 3.3 V dengan konsumsi arus sebesar 250mA untuk pengiriman data (Tx) dan sebesar 50mA untuk penerimaan data (Rx). Konsumsi tegangan dan arus yang kecil menjadi salah satu keunggulan modul wireless ini dibandingkan dengan modul wireless lainnya PIN yang terdapat pada XBee PRO sebanyak 20 pin namun yang digunakan hanya 6 pin saja yaitu Vcc, Gnd, Reset, Dout (Tx), Din(Rx) dan PWMO/RSSI (Indikator). Prinsip kerja dan penerimaan data dari modul Wireless XBee PRO dapat dilihat pada Gambar 3.5.

Gambar 3.4. XBeePRO.



Gambar 3.5. Ilustrasi Prinsip Kerja Modul XBee PRO.

Pada robot terpasang ATMega16 sebagai kendali kontrol untuk motor. Motor yang digunakan pada robot adalah motor DC. Penampang fisik ATMega16 ditunjukkan pada Gambar 3.6. Sedangkan bentuk fisik motor DC ditunjukkan pada 3.7.

Gambar 3.6. Mikrokontroler ATMega16.

Gambar 3.7. Motor DC.

Untuk sumber daya pada robot, digunakan baterai Li-Po. Baterai Li-Po (Lithium Polymer) adalah salah satu jenis baterai yang rechargeable yang sering digunakan pada robot, mobil RC, helicopter dan quadcopter. Baterai Li-Po tersusun atas sel baterai yang memiliki tegangan sebesar 3.7 Volt dengan voltase maksimum sebesar 4.2 Volt dan voltase minimum sebesar 3.5 Volt tiap selnya. Jika pada baterai Li-Po tertulis keterangan 4s artinya baterai Li-Po tersebut memiliki 4 sel baterai dengan tegangan sebesar 14.8 Volt. Contoh bentuk fisik dari baterai Li-Po ditunjukkan pada Gambar 3.8.

Gambar 3.8. Baterai Li-Po.

3.6 Perancangan Kode Program

Pada perancangan kode program berikut dijelaskan mengenai perancangan beberapa fungsi – fungsi pada sistem yang dijabarkan dalam bentuk *pseudocode.*

3.6.1 Fungsi Pemilihan Data *Training*

Pengambilan sekuen atau urutan data *training* yang digunakan pada sistem ditentukan secara acak dengan lama perekaman data (jumlah *record*) yang dapat ditentukan oleh pengguna. Lama perekaman bervariasi antara 5 detik, 10 detik, hingga 30 detik. *Psudocode* fungsi pemilihan data training dapat dilihat pada Gambar 3.9.

```
SetRawDataTrain(rawData, timeRecord)
        double[][] data
        lengthData = timeRecord * 128
        index = random from 0 to 128* total rawData
        while index+lenghtData > total rawData do
                index = random from 0 to 128* total rawData
        for i = 0 to total rawData do
                for j = 0 to 14 do
                        data[i][j] = rawData[index+i][j]
                end for
        end for

        return data
```

Gambar 3.9. *Pseudocode* pemilihan data training.

3.6.2 Fungsi Normalisasi Data

Normalisasi data dilakukan untuk setiap data dengan lama waktu 0.5 detik (64 *record*). Proses normalisasi data dilakukan dengan memusatkan data menuju ke 0. Hal ini dilakukan dengan cara mengurangi nilai setiap data pada satu fitur dengan rata – rata nilai dari fitur tersebut. Proses ini dilakukan secara terpisah untuk setiap fitur. Kemudian setiap nilai pada data dibagi oleh 30 yang merupakan rata-rata dari amplitudo sinyal sehingga data berada pada kisaran rentang -1 dan 1. *Pseudocode* fungsi normalisasi data dapat dilihat pada Gambar 3.10.

```
NormalizeData(data)
        count = 0
        double[][] result
        double[] sum
        for i=0 to total.data do
                for j=0 to 14 do
                        sum[j]+=data[i][j]
                count++
                if count = 63 then
                        count=0
                        for k=0 to i do
                                for l=0 to 14 do
                                        result[k][l]         =
(data[k][l] - sum[l]/64)/30
                                end for
                        end for
                end if
                end for
        end for
        return result
```

Gambar 3.10. *Pseudocode* normalisasi data.

3.6.3 Fungsi Klasifikasi

```
int Model.Classify(input)
TestData(dataInput)
        int count = 0
        int[] totalOutput
        int result
        for i=0 to 64 do
                count++
                result = Model.Classify(dataInput[i])
                totalOutput[result]++
        result = get index max(totalOutput)
        return result
```

Gambar 3.11. *Pseudocode* proses klasifikasi perintah gerak.

Kelas gerak didapatkan dari kelas terbanyak yang dihasilkan oleh proses klasifikasi data EEG selama 0.5 detik (64 *record*). Proses klasifikasi dilakukan dengan menggunakan model yang telah dilakukan proses *training* sebelumnya. Data yang digunakan untuk proses klasifikasi telah melewati *preproccess* sebelumnya (*band pass filter* dan normalisasi data). *Pseudocode* fungsi klasifikasi ditunjukkan pada Gambar 3.11.

3.6.3 Fungsi Kirim Perintah Gerak

Setelah kelas perintah gerak diperoleh dari proses klasifikasi, maka perintah gerak perlu dikirimkan kepada robot agar dapat dijalankan. Pengiriman perintah gerak dilakukan melalui komunikasi serial data. Karakter tertentu dikirimkan sesuai dengan perintah gerak yang didapatkan. *Pseudocode* pengiriman perintah gerak robot ditunjukkan pada Gambar 3.12

```
SendCommandToRobot(kelasGerak)
       if kelasGerak is 'maju' then
              send('W')
       else if kelasGerak is 'mundur' then
              send('X')
       else if kelasGerak is 'kanan' then
              send('D')
       else if kelasGerak is 'kiri' then
              send('A')
       else if kelasGerak is 'netral' then
              send('S')
       end if
```

Gambar 3.12. *Pseudocode* pengiriman perintah gerak robot.

[Halaman ini sengaja dikosongkan]

BAB IV
IMPLEMENTASI

Pada bab ini akan dibahas mengenai implementasi yang dilakukan berdasarkan rancangan yang telah dijabarkan pada bab sebelumnya. Sebelum penjelasan implementasi akan ditunjukkan terlebih dahulu lingkungan untuk melakukan implementasi.

4.1 Lingkungan Implementasi

Dalam membangun aplikasi ini digunakan beberapa perangkat pendukung baik perangkat keras maupun perangkat lunak. Spesifikasi perangkat keras dan perangkat lunak yang digunakan dalam implementasi ini ditampilkan pada Tabel 4.1.

Tabel 4.1. Lingkungan implementasi perangkat.

Perangkat	Spesifikasi
Perangkat Keras	Prossesor: Intel® Core(TM) i5-255 CPU @ 2.14 GHz Memori: 4.00 GB
Perangkat Lunak	Sistem Operasi: Windows 8.1 Pro 64-bit Perangkat pengembang: - Microsoft Visual Studio Professional 2013 Version 12.0.30501.00 Update 2 - Microsoft .NET Framework Version 4.5.50710 Perangkat Lunak Pendukung: - Emotiv Xavier Controlpanel PREMIUM 3.0.0.40 - Eloquera Database - Notepad++ v6.6.8 Pustaka: Accord, EmoClient, Visual Wrapper
Perangkat Robot	Motor penggerak: Motor DC 12V Sumber arus: Baterai Li-Po 2200 mAh Mikrokontoler: ATmega16 Koneksi nirkabel: XBee PRO Series 1

4.2 Implementasi *Brain Computer Interface* (BCI)

Pada subbab ini dijelaskan implementasi BCI beserta fungsi-fungsi yang ada di dalamanya. Terdapat tiga bagian utama dari BCI yang dibuat, diantaranya Emotiv Engine Realtime, Configuration Data Model dan Create Model.

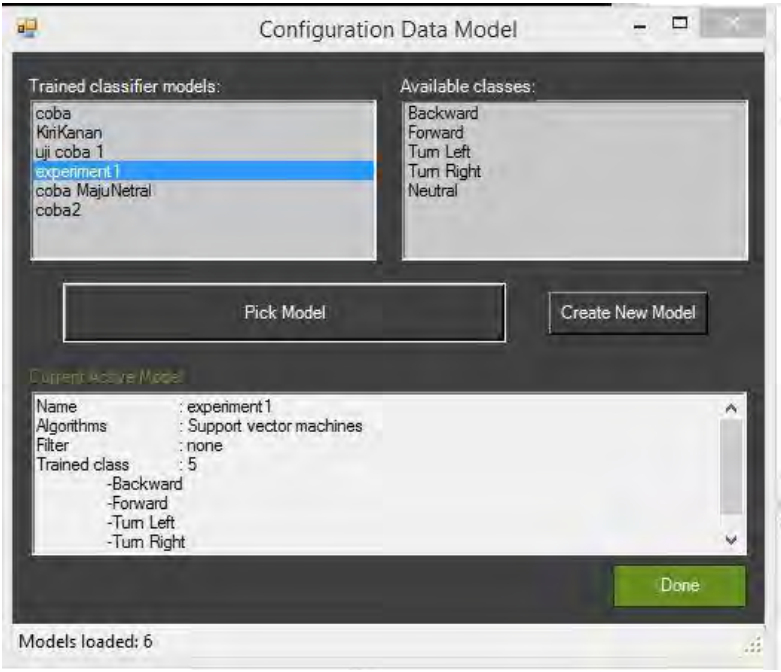
4.2.1 Emotiv Engine Realtime

Emotiv Engine Realtime berfungsi untuk menampilkan data gelombang EEG secara realtime dan menampilkan status dari neuro headset. Selain itu juga terdapat tombol kontrol untuk kontrol robot dan *connection port* untuk koneksi robot dengan BCI. Tampilan antarmuka Emotiv Engine Realtime dapat dilihat pada Gambar 4.1. Untuk implementasi code secara lengkap terdapat pada LAMPIRAN B.

Gambar 4.1. Tampilan antarmuka Emotiv Engine Realtime.

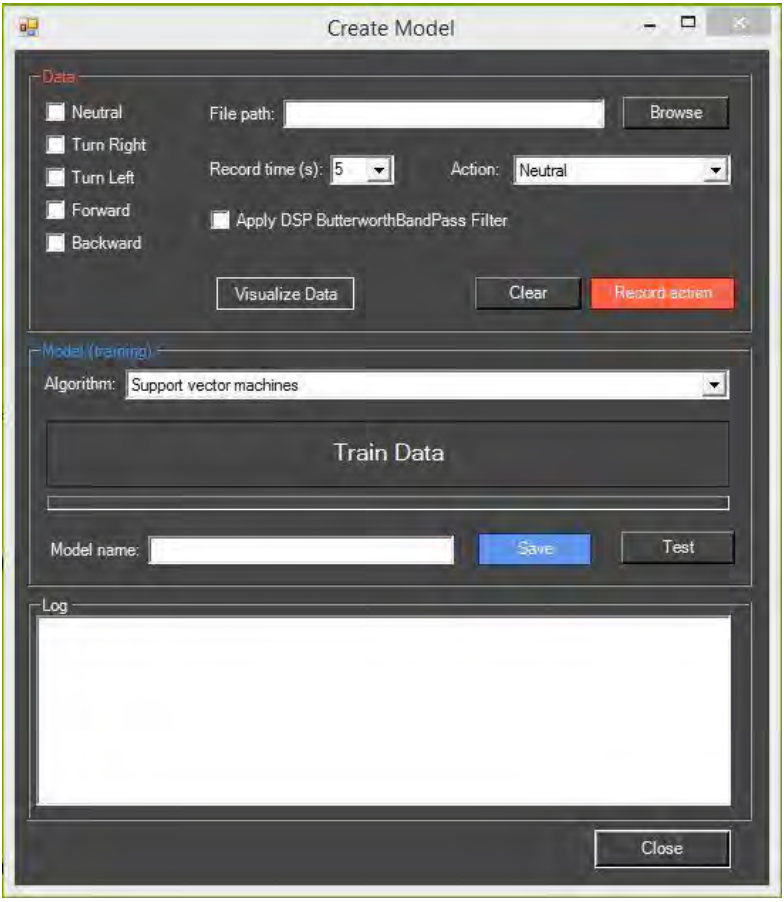
4.2.2 Configuration Data Model

Configuration Data Model berfungsi untuk memilih data model yang ingin digunakan. Selain itu juga ditampilkan daftar data model yang telah dibuat sebelumnya saat melakukan proses *training.* Ditampilkan juga kelas-kelas yang tersedia pada data model. Tampilan antarmuka *form* Configuration Data Model dapat dilihat pada Gambar 4.2. Untuk implementasi code secara lengkap terdapat pada LAMPIRAN B.



Gambar 4.2. Tampilan antarmuka form Configuration Data Model.

4.2.3 Create Model



Gambar 4.3. Tampilan antarmuka form Create Model.

Pada *form* Create Model, terdapat pilihan proses pembacaan *raw* data EEG, *training* dan *testing* data, serta pilihan untuk menyimpan model yang telah dibuat. Tersedia beberapa pilihan parameter yang dapat dipilih diantaranya: *record time, Apply DSP*

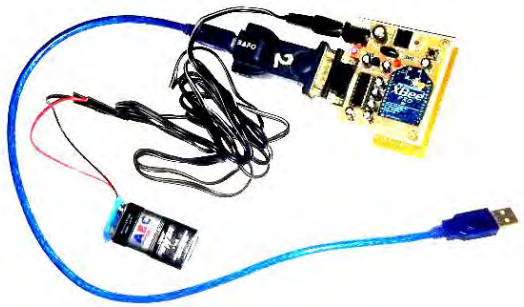
ButterworthBandPass Filter, dan algoritma yang digunakan. Tampilan antarmuka *form* Create Model dapat dilihat pada Gambar 4.3. Untuk implementasi code secara lengkap terdapat pada LAMPIRAN B.

4.3 Implementasi Robot

Pada subbab ini dijelaskan proses pembuatan bagian-bagian Robot. Implementasi pada robot dapat dipisahkan menjadi dua bagian utama: *transmitter* dan tubuh robot.

4.3.1 Implementasi *Transmitter* Robot

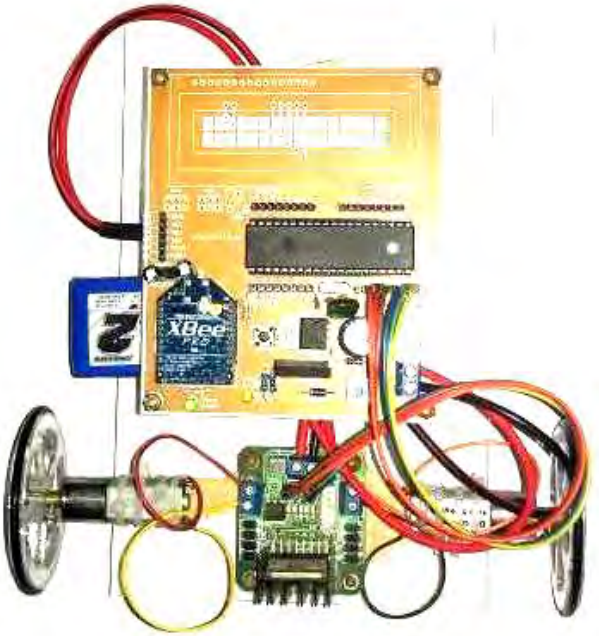
Transmitter berfungsi untuk mengirimkan dan menerima data dari BCI ke robot. Rangkaian pada *transmitter* menggunakan komponen utama XBee PRO untuk koneksi secara nirkabel. Komunikasi yang digunakan adalah komunikasi serial data. Baterai 9V digunakan sebagai pembangkit daya *transmitter*. Hasil implementasi *transmitter* ditunjukkan pada Gambar 4.4.



Gambar 4.4. Transmitter XBee PRO.

4.3.2 Implementasi Tubuh Robot

Robot yang dibuat adalah robot beroda yang dapat dikendalikan secara nirkabel serta dapat bergerak maju, mundur, belok kanan, dan belok kiri. Kontrol gerak pada robot ini diproses di program mikrokontroler. Hasil implementasi tubuh robot ditunjukkan pada Gambar 4.5. Untuk implementasi kode program mikrokontroler pada robot dapat dilihat di LAMPIRAN A.



Gambar 4.5. Tampak atas robot beroda.

BAB V
UJI COBA DAN EVALUASI

Pada bab ini dijelaskan uji coba yang dilakukan pada aplikasi yang telah dikerjakan serta analisa dari uji coba yang telah dilakukan. Pembahasan pengujian meliputi lingkungan uji coba, skenario uji coba yang meliputi uji kebenaran dan uji kinerja serta analisa setiap pengujian.

5.1 Lingkungan Uji Coba

Spesifikasi perangkat keras dan perangkat lunak yang digunakan dalam tahap uji coba ditampilkan pada Tabel 5.

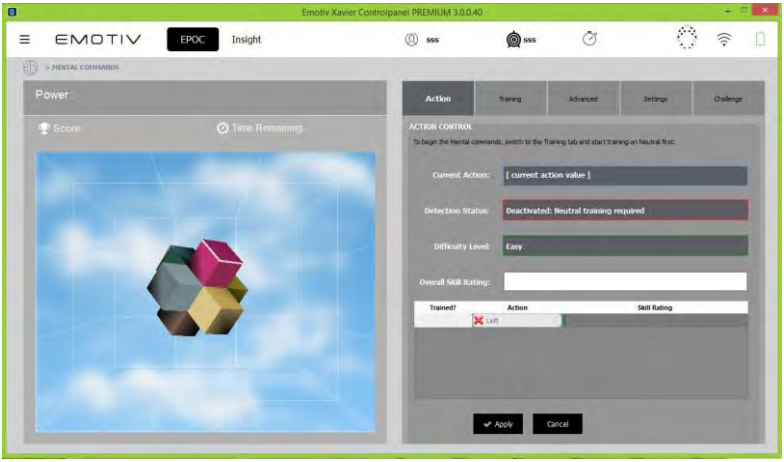
Tabel 5.1. Lingkungan uji coba perangkat lunak

Perangkat	Spesifikasi
Perangkat Keras	Prossesor: Intel® Core(TM) i5-255 CPU @ 2.14 GHz Memori: 4.00 GB
Perangkat Lunak	Sistem Operasi: Windows 8.1 Pro 64-bit Perangkat pengembang: - Microsoft Visual Studio Professional 2013 Version 12.0.30501.00 Update 2 - Microsoft .NET Framework Version 4.5.50710 Perangkat Lunak Pendukung: - Emotiv Xavier Controlpanel PREMIUM 3.0.0.40 - Eloquera Database - Notepad++ v6.6.8 Pustaka: Accord, EmoClient, Visual Wrapper
Perangkat Robot	Motor penggerak: Motor DC 12V Sumber arus: Baterai Li-Po 2200 mAh Mikrokontoler: ATmega16 Koneksi nirkabel: XBee PRO Series 1

5.2 Data *Training*

Proses perekaman data *raw* EEG merupakan proses yang sangat penting mengingat semakin bagus kualitas data EEG yang dihasilkan maka semakin bagus pula model yang dihasilkan. Perekaman data EEG yang berkualitas dihasilkan ketika kondisi subjek uji rileks dan konsentrasi pikiran subjek uji fokus pada salah satu kelas perintah dari lima kelas perintah yang ada. Lima kelas perintah itu adalah netral, maju, mundur, belok kanan, dan belok kiri.

Untuk membantu subjek uji berada pada kondisi fokus digunakan visualisasi *animate model box* yang dapat bergerak ke atas, ke bawah, ke kiri dan ke kanan. Visualisasi model *box* ini terdapat pada Mental Command Detection Suite yang merupakan perangkat lunak bawaan dari Emotiv seperti yang tampak pada Gambar 5.1.



Gambar 5.1. Mental command detection suite.

Sebelumnya, subjek uji harus melakukan proses *training action control* terlebih dahulu untuk dapat menggerakan visualisasi model box. Setiap sesi *training* berlangsung selama 8 detik. Selama

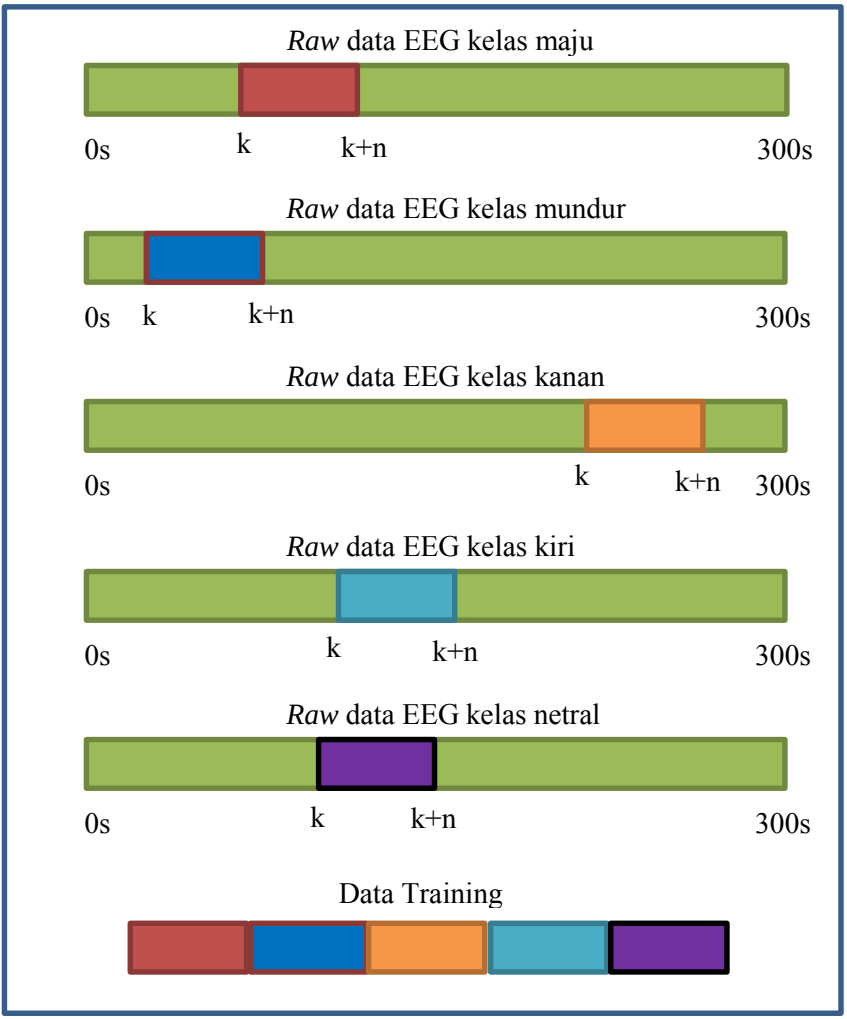
proses *training*, subjek uji harus rileks, tidak boleh banyak bergerak dan tidak boleh mengedipkan mata. Berikut penjelasan lengkap proses training *action control* pada Mental Command Detection Suite.

1) *Action neutral*
   Subjek uji dalam kondisi rileks, tenang, dan tidak perlu memikirkan apapun.
1) *Action lift*
   Subjek uji dalam kondisi rileks dan fokus membayangkan *model box* bergerak ke atas.
2) *Action drop*
   Subjek uji dalam kondisi rileks dan fokus membayangkan *model box* bergerak ke bawah.
3) *Action left*
   Subjek uji dalam kondisi rileks dan fokus membayangkan *model box* bergerak ke kiri.
4) *Action right*
   Subjek uji dalam kondisi rileks dan fokus membayangkan *model box* bergerak ke kanan.

Setelah selesai melakukan proses training pada Mental Command Detection Suite, subjek uji dapat menggerakkan *model box* dengan menggunakan pikiran. Misal, apabila subjek uji fokus membayangkan *model box* bergerak ke kiri, maka *model box* pun bergeser bergerak ke kiri. Begitu juga dengan action control yang lainnya. Semakin fokus subjek uji membayangkan gerakan, maka semakin kuat indikator *power* yang ditunjukkan dan semakin kuat serta stabil pula pergerakan dari *model box.*

Seringkali subjek uji amatir melakukan kesalahan dengan mengejangkan otot, mengernyitkan dahi atau menahan nafas pada saat melakukan *mental command*. Tentu saja hal ini tidak dibenarkan karena mengakibatkan data EEG yang dihasilkan menjadi tidak akurat. Hal ini dikarenakan sinyal otot merupakan *noise* dalam data EEG. Tujuan yang diharapkan adalah benar-benar *pure mental command.* Untuk membiasakan diri dengan *mental*

command, pada Mental Command Detection Suite tersedia level dan score yang dapat ditingkatkan. Level dan score ini menunjukkan tingkat keahlian subjek uji dalam melakukan *mental command*.



Gambar 5.2. Visualisasi pemilihan data training.

Pada saat proses perekaman, subjek uji melakukan *mental command* dengan bantuan visualisasi *model box* pada Mental Command Detection Suite. Agar data EEG berkualitas bagus, perekaman data EEG dilakukan ketika *model box* bergerak secara stabil sesuai dengan *mental command* yang dilakukan subjek uji. Untuk masing – masing kelas perintah diperoleh secara akumulasi 5 menit *raw* Data EEG. *Raw* Data EEG inilah yang sebagian digunakan sebagai data *training* dan seluruhnya digunakan sebagai data *testing*.

Untuk pemilihan data *training,* dilakukan pemilihan sekuen *raw* data EEG secara acak dengan panjang sekuen n detik dari setiap kelas perintah. Selanjutnya sekuen *raw* data EEG yang telah didapatkan dari masing-masing kelas perintah digabung sebagai data training. Visualisasi pemilihan data training dapat dilihat pada Gambar 5.2.

5.3 Skenario dan Evaluasi Pengujian

Uji coba ini dilakukan untuk menguji apakah fungsionalitas program telah diimplementasikan dengan benar dan berjalan sebagaimana mestinya. Uji coba akan didasarkan pada beberapa skenario untuk menguji kesesuaian dan kinerja aplikasi.

Skenario pengujian terdiri dari 5 pengujian yaitu:

1. Skenario perhitungan akurasi dan *power* metode gabungan LDA dan SVM tanpa *band pass filter* dengan panjang sekuen *raw* data *training* 30 detik dengan menggunakan 5 kelas gerak.
2. Skenario perhitungan akurasi dan *power* metode gabungan LDA dan SVM dengan *band pass filter* dengan panjang sekuen *raw* data *training* 30 detik dengan menggunakan 5 kelas gerak.
3. Skenario perhitungan akurasi dan *power* metode gabungan LDA dan SVM dengan *band pass filter* dengan variasi panjang sekuen *raw* data *training* dengan menggunakan 5 kelas gerak.

4. Skenario perhitungan akurasi dan *power* metode gabungan LDA dan SVM dengan *band pass filter* dengan batas ambang bervariasi dengan panjang sekuen *raw* data *training* 30 detik dengan menggunakan 5 kelas gerak.
5. Skenario perhitungan akurasi dan *power* metode gabungan LDA dan SVM dengan *band pass filter* dengan panjang sekuen *raw* data *training* 30 detik dengan menggunakan variasi jumlah kelas gerak.
6. Skenario perhitungan akurasi metode LDA, SVM, dan gabungan LDA dan SVM, dengan *band pass filter* dengan panjang sekuen *raw* data *training* 30 detik dengan menggunakan 5 kelas gerak.

Untuk evaluasi pengujian digunakan 2 parameter uji, yaitu akurasi dan *power.* Akurasi dihitung dengan rumus:

$$\boldsymbol{Akurasi} = \frac{\boldsymbol{True\ Positive}}{\boldsymbol{Jumlah\ seluruh\ data}} \boldsymbol{x}\ \mathbf{100}\% \qquad (5.1)$$

Sedangkan *power* dihitung dari jumlah sekuen *True Positive* terbanyak selain kelas netral yang berhasil diperoleh.

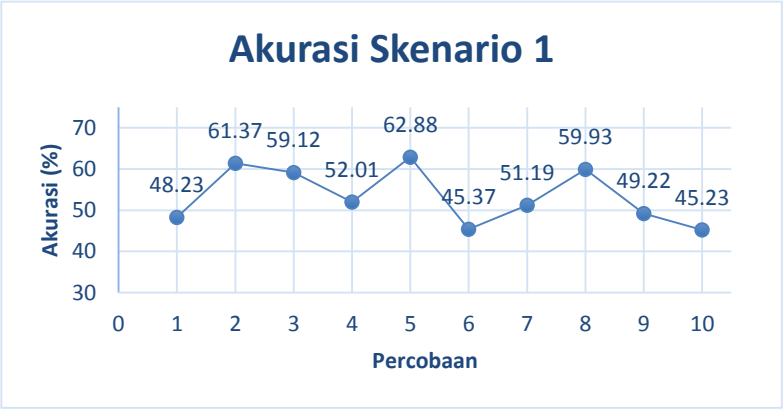
5.3.1 Uji Coba Skenario 1

Skenario uji coba 1 adalah perhitungan akurasi dan *power* metode gabungan LDA dan SVM tanpa *band pass filter* dengan panjang sekuen *raw* data *training* 30 detik dengan menggunakan 5 kelas gerak.

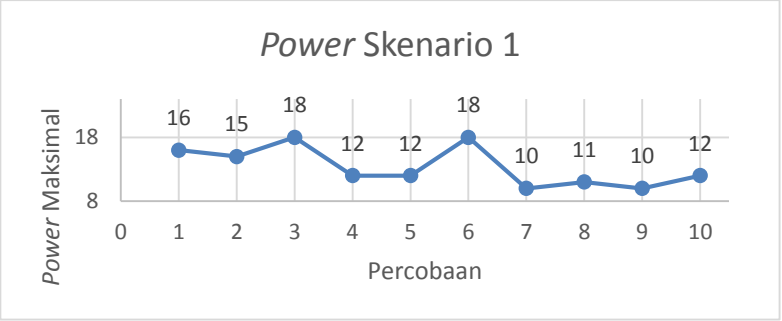
Setelah dilakukan percobaan, maka harus dilakukan evaluasi untuk melihat sejauh mana kesesuaian program dengan data uji yang disediakan. Untuk menentukan kesesuaian keluaran dengan program dan analisa yang telah dibuat diperlukan pengujian akurasi dan *power*. Perhitungan akurasi dilakukan menggunakan persamaan 5.1. Gambar 5.3. Adalah grafik hasil

akurasi rata – rata uji coba skenario 1. Grafik *power* uji coba skenario 1 dapat dilihat pada Gambar 5.4.

Pada akhir proses, dalam 10 kali proses percobaan, didapatkan rata-rata nilai akurasi sebesar 53.45% dengan *power* maksimal mampu mencapai 18. Hasil uji coba skenario 1 secara lengkap dapat dilihat pada LAMPIRAN C.



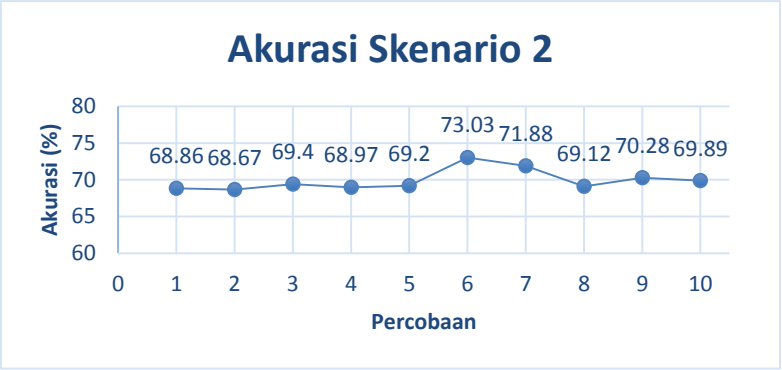
Gambar 5.3. Grafik akurasi untuk skenario 1.



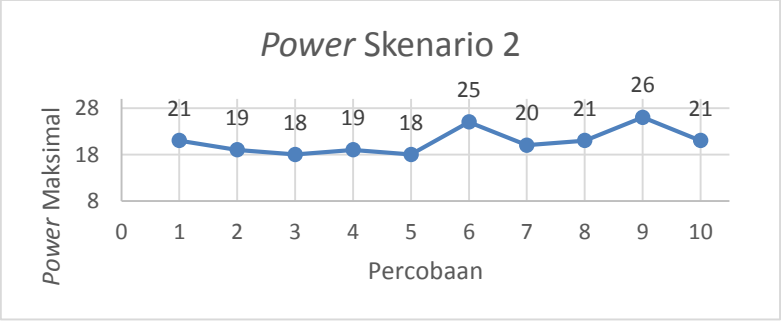
Gambar 5.4. Grafik *power* maksimal untuk skenario 1.

5.3.2 Uji Coba Skenario 2

Skenario uji coba 2 adalah perhitungan akurasi dan *power* metode gabungan LDA dan SVM dengan band pass filter dengan panjang sekuen raw data training 30 detik dengan menggunakan 5 kelas gerak. *Band pass filter* yang digunakan adalah Butterworth *band pass filter* order 4 dengan ambang batas bawah = 29 dan atas = 40.



Gambar 5.5. Grafik akurasi untuk skenario 2.



Gambar 5.6. Grafik *power* maksimal untuk skenario 2.

Setelah dilakukan percobaan, maka harus dilakukan evaluasi untuk melihat sejauh mana kesesuaian program dengan data uji yang disediakan. Untuk menentukan kesesuaian keluaran dengan program dan analisa yang telah dibuat diperlukan pengujian akurasi dan *Power*. Perhitungan akurasi dilakukan menggunakan persamaan 5.1. Gambar 5.5 adalah grafik hasil akurasi rata – rata uji coba skenario 2. Grafik *power* uji coba skenario 2 dapat dilihat pada Gambar 5.6

Pada akhir proses, dalam 10 kali proses percobaan, didapatkan rata-rata nilai akurasi sebesar 69.90% dengan *power* maksimal mampu mencapai 26. Hasil uji coba skenario 2 secara lengkap dapat dilihat pada LAMPIRAN C.

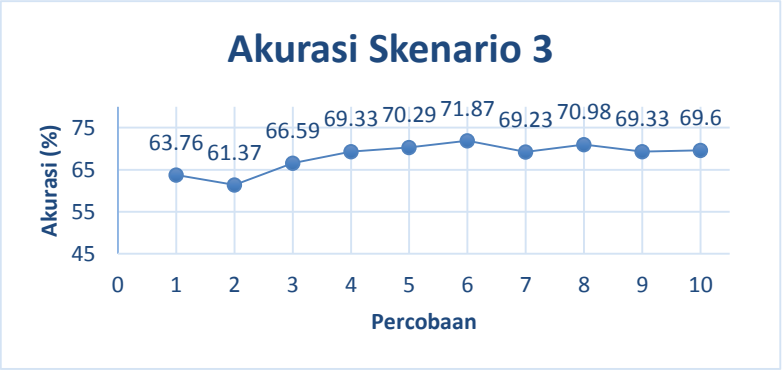
5.3.3 Skenario Uji Coba 3

Tabel 5.2. Variasi lama perekaman data *training*

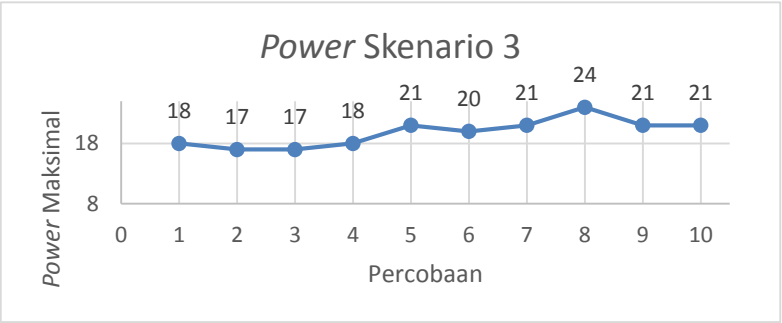
Percobaan	Lama Perekaman (detik)
1	5
2	8
3	10
4	15
5	20
6	25
7	30
8	40
9	50
10	60

Skenario uji coba 3 adalah perhitungan akurasi dan power metode gabungan LDA dan SVM dengan band pass filter dengan

variasi panjang sekuen *raw* data *training* dengan menggunakan 5 kelas gerak. *Band pass filter* yang digunakan adalah Butterworth band pass filter order 4 dengan ambang batas bawah = 29 dan atas = 40. Variasi panjang sekuen (lama waktu perekaman) *raw* data *training* ditunjukkan pada Tabel 5.2.



Gambar 5.7. Grafik akurasi untuk skenario 3.



Gambar 5.8. Grafik *power* maksimal untuk skenario 3.

Setelah dilakukan percobaan, maka harus dilakukan evaluasi untuk melihat sejauh mana kesesuaian program dengan data uji yang disediakan. Untuk menentukan kesesuaian keluaran dengan program dan analisa yang telah dibuat diperlukan

pengujian akurasi dan *Power*. Perhitungan akurasi dilakukan menggunakan persamaan 5.1. Gambar 5.7 adalah grafik hasil akurasi rata – rata uji coba skenario 3. Grafik *power* uji coba skenario 3 dapat dilihat pada Gambar 5.8.

Pada akhir proses, dalam 10 kali proses percobaan, didapatkan nilai akurasi terbesar 71.87% dari percobaan ke-6, yaitu dengan menggunakan lama waktu perekaman 25 detik dan *power* maksimal mencapai 24 dari percobaan ke-8, yaitu dengan menggunakan lama waktu perekeman 40 detik. Hasil uji coba skenario 3 secara lengkap dapat dilihat pada LAMPIRAN C.

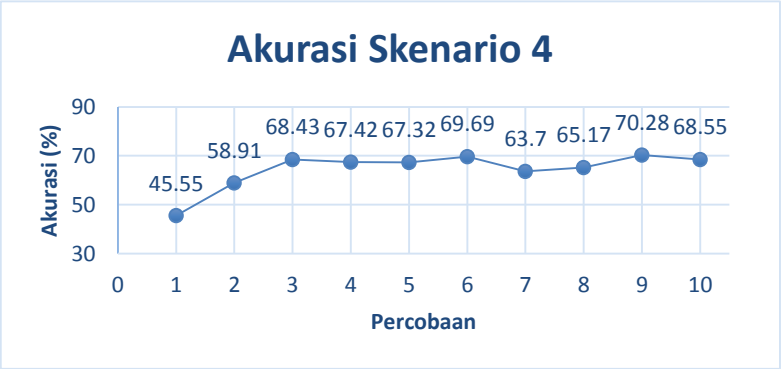
5.3.4 Skenario Uji Coba 4

Tabel 5.3. Variasi nilai ambang batas butterworth band pass filter order 4 yang diuji coba.

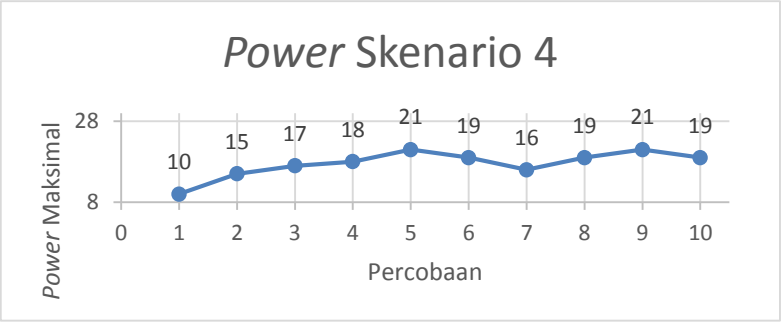
Percobaan	Batas Bawah	Batas Atas
1	16	40
2	22	40
3	31	40
4	33	40
5	29	37
6	29	39
7	29	45
8	29	43
9	28	39
10	30	40

Skenario uji coba 4 adalah perhitungan akurasi dan power metode gabungan LDA dan SVM dengan *band pass filter* dengan batas ambang bervariasi dengan panjang sekuen raw data training 30 detik dengan menggunakan 5 kelas gerak. *Band pass filter* yang

digunakan adalah Butterworth *band pass filter* order 4 dengan nilai ambang batas bervariasi. Variasi nilai ambang batas yang diberikan ditunjukkan pada Tabel 5.3.



Gambar 5.9. Grafik akurasi untuk skenario 4.



Gambar 5.10. Grafik *power* maksimal untuk skenario 4.

Setelah dilakukan percobaan, maka harus dilakukan evaluasi untuk melihat sejauh mana kesesuaian program dengan data uji yang disediakan. Untuk menentukan kesesuaian keluaran dengan program dan analisa yang telah dibuat diperlukan pengujian akurasi dan *Power*. Perhitungan akurasi dilakukan menggunakan persamaan 5.3. Gambar 5.9 adalah grafik hasil

akurasi rata – rata uji coba skenario 4. Grafik *power* uji coba skenario 4 dapat dilihat pada Gambar 5.10.

Pada akhir proses, dalam 10 kali proses percobaan, didapatkan nilai akurasi terbaik 70.28% dari percobaan ke-9, yaitu dengan menggunakan ambang batas bawah = 28 dan batas atas = 39 dengan *power* maksimal mampu mencapai 21. Hasil uji coba skenario 4 secara lengkap dapat dilihat pada LAMPIRAN C.

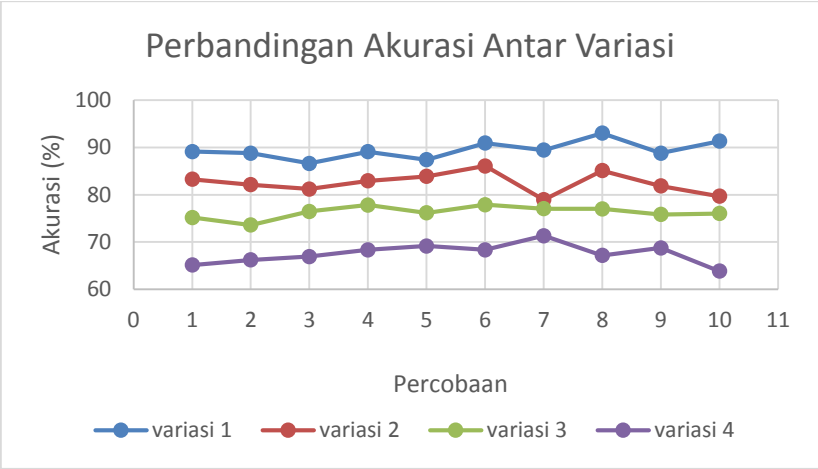
5.3.5 Skenario Uji Coba 5

Skenario uji coba 5 adalah perhitungan akurasi dan *power* metode gabungan LDA dan SVM dengan band pass filter dengan panjang sekuen raw data training 30 detik dengan menggunakan variasi jumlah kelas gerak. Variasi jumlah kelas gerak yang diberikan dapat dilihat pada Tabel 5.4.

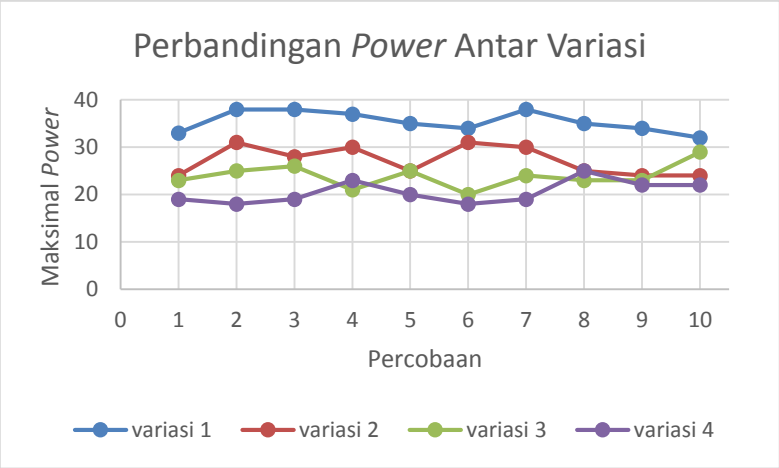
Tabel 5.4. Variasi jumlah kelas gerak yang diuji coba.

Variasi ke -	Jumlah Kelas Gerak	Jenis Kelas Gerak
1	2	Netral, Maju
2	3	Netral, Maju, Mundur
3	4	Netral, Maju, Kanan, Kiri
4	5	Netral, Maju, Mundur, Kanan, Kiri

Setelah dilakukan percobaan, maka harus dilakukan evaluasi untuk melihat sejauh mana kesesuaian program dengan data uji yang disediakan. Untuk menentukan kesesuaian keluaran dengan program dan analisa yang telah dibuat diperlukan pengujian akurasi dan *Power*. Perhitungan akurasi dilakukan menggunakan persamaan 5.1. Gambar 5.11 adalah grafik perbandingan hasil akurasi rata – rata antar variasi uji coba skenario 5. Grafik perbandingan *power* uji coba skenario 2 antar variasi dapat dilihat pada Gambar 5.12.



Gambar 5.11. Perbandingan akurasi antar variasi.

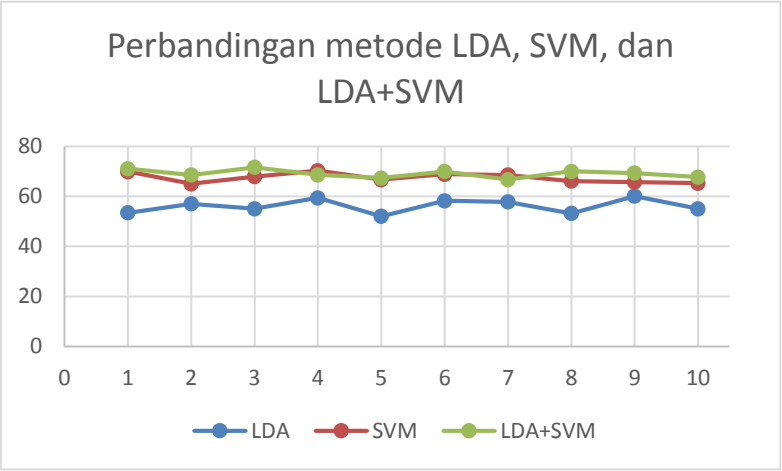


Gambar 5.12. Perbandingan *power* maksimal antar variasi.

Pada akhir proses, dalam 10 kali proses percobaan, didapatkan nilai rata-rata akurasi untuk variasi 1 sebesar 89.45%, variasi 2 sebesar 82.50%, variasi 3 sebesar 76.30% dan variasi 4 sebesar 67.52%. Hasil uji coba skenario 5 secara lengkap dapat dilihat pada LAMPIRAN C.

5.3.6 Skenario Uji Coba 6

Skenario uji coba 6 adalah perhitungan akurasi metode LDA, SVM, dan gabungan LDA dan SVM, dengan band pass filter dengan panjang sekuen raw data training 30 detik dengan menggunakan 5 kelas gerak.



Gambar 5.13. Perbandingan rata-rata akurasi metode LDA, SVM, dan LDA+SVM.

Setelah dilakukan percobaan, maka harus dilakukan evaluasi untuk melihat sejauh mana kesesuaian program dengan data uji yang disediakan. Untuk menentukan kesesuaian keluaran dengan program dan analisa yang telah dibuat diperlukan pengujian akurasi dan *Power*. Perhitungan akurasi dilakukan menggunakan

persamaan 5.1. Gambar 5.13 adalah grafik perbandingan hasil akurasi rata – rata antar variasi uji coba skenario 6.

Pada akhir proses, dalam 10 kali proses percobaan, didapatkan nilai rata-rata akurasi untuk metode LDA sebesar 56.132%, metode SVM sebesar 67.42% dan metode gabungan LDA dan SVM sebesar 69.063%.

5.4 Analisis Hasil Uji Coba

Setelah dihitung rata-rata akurasi, pertanyaan yang sering muncul adalah apakah nilai rata-rata akurasi tersebut betul-betul representasi data yang baik dari akurasi hasil percobaan yang telah dilakukan. Oleh karena itu perlu pula dilakukan perhitungan standar deviasi untuk mengetahui rentang atau variasi hasil. Adapun rumus untuk menghitung standar deviasi adalah sebagai berikut:

$$S = \sqrt{\frac{\sum(Xi - mean)^2}{n - 1}} \tag{5.2}$$

dimana *S* adalah standar deviasi, *mean* adalah rata-rata data, dan *n* adalah jumlah data. Standar deviasi dapat menggambarkan seberapa jauh bervariasinya data. Jika nilai standar deviasi jauh lebih besar dari *mean,* maka nilai *mean* kurang merepresentasikan performa untuk semua data. Sedangkan jika nilai standar deviasi sangat kecil dibandingkan dengan *mean,* maka nilai mean dapat digunakan sebagai representasi dari keseluruhan data.

5.4.1 Analisis Hasil Uji Coba Skenario 1 dan 2

Berdasarkan pada nilai standar deviasi dari hasil akurasi masing-masing skenario yang ditunjukkan pada Tabel 5.5, nilai standar deviasi yang dihasilkan sangat jauh lebih kecil daripada *mean.* Hal itu menunjukkan bahwa dalam kasus ini, *mean* merupakan representasi yang baik dari hasil percobaan. Selain itu, standar deviasi juga menyimpan informasi kefluktuatifan data. Semakin tinggi nilai standar deviasi, maka dapat dikatakan data

semakin fluktuatif. Di antara skenario 1 dan skenario 2 yang diujicobakan, maka tingkat akurasi pada skenario uji coba 1 lebih fluktuatif jika dibandingkan skenario 2. Selain itu nilai rata-rata akurasi yang dihasilkan skenario 1 lebih rendah dibandingkan dengan skenario 2.

Tabel 5.5. Perbandingan standar deviasi dan mean akurasi uji coba skenario 1 dan 2.

	Uji Coba Skenario 1	Uji Coba Skenario 2
Mean Akurasi	53.46%	66.41%
Standar Deviasi Akurasi	6.76	3.86

Jika dilihat dari perbandingan rata-rata akurasi dan nilai *power* maksimal yang dihasilkan dari uji coba skenario 1 dan uji coba skenario 2, maka dapat dikatakan bahwa penggunaan band pass filter untuk *preprocess* dapat meningkatkan akurasi rata-rata.

5.4.2 Analisis Hasil Uji Coba Skenario 3

Dilihat dari perbandingan rata-rata akurasi dan nilai *power* maksimal yang dihasilkan dari uji coba skenario 3, Semakin lama waktu perekeman data, akurasi yang dihasilkan semakin baik. Namun setelah melewati lama waktu rekaman 20 detik, akurasi yang dihasilkan relatif tidak ada perubahan.

5.4.3 Analisis Hasil Uji Coba Skenario 4

Dilihat dari perbandingan rata-rata akurasi dan nilai *power* maksimal yang dihasilkan dari uji coba skenario 4 didapatkan bahwa ambang batas yang baik untuk ambang batas bawah berkisar antara 28-30 dan ambang batas atas berkisar antara 38-43.

5.4.4 Analisis Hasil Uji Coba Skenario 5

Dilihat dari uji coba skenario 5 didapatkan bahwa semakin sedikit kelas gerak, maka akurasi yang dihasilkan semakin tinggi. Akurasi rata-rata untuk model yang menggunakan 2 kelas gerak, yaitu kelas gerak netral dan kelas gerak maju dapat mencapai 89.45%.

5.4.5 Analisis Hasil Uji Coba Skenario 6

Berdasarkan pada nilai standar deviasi dari hasil akurasi masing-masing skenario yang ditunjukkan pada Tabel 5.6, nilai standar deviasi yang dihasilkan sangat jauh lebih kecil daripada *mean.* Hal itu menunjukkan bahwa dalam kasus ini, *mean* merupakan representasi yang baik dari hasil percobaan. Selain itu, standar deviasi juga menyimpan informasi kefluktuatifan data. Semakin tinggi nilai standar deviasi, maka dapat dikatakan data semakin fluktuatif.

Dilihat dari rata-rata akurasi, metode klasifikasi SVM lebih baik dibandingkan dengan metode LDA. Sedangkan metode gabungan LDA + SVM mampu menghasilkan tingkat akurasi yang paling baik jika dibandingkan dengan metode LDA atau SVM, yaitu 69.06%.

Tabel 5.6. Perbandingan standar deviasi dan mean akurasi metode LDA, SVM, dan LDA + SVM

	LDA	SVM	LDA + SVM
Mean Akurasi	56.132%	67.42%	69.06%
Standar Deviasi Akurasi	2.79	1.92	1.59

BAB VI
KESIMPULAN DAN SARAN

Bab ini membahas mengenai kesimpulan yang dapat diambil dari hasil uji coba yang telah dilakukan sebagai jawaban dari rumusan masalah yang dikemukakan. Selain kesimpulan, juga terdapat saran yang ditujukan untuk pengembangan penelitian lebih lanjut.

6.1 Kesimpulan

Dari hasil uji coba yang telah dilakukan terhadap pembuatan model, dapat diambil kesimpulan sebagai berikut:

1. Metode gabungan LDA dan SVM dapat menghasilkan model yang baik untuk pengklasifikasian *Mental Command* data EEG untuk kontrol gerak robot.
2. Metode gabungan LDA dan SVM dengan menggunakan Butterworth *band pass filter* mampu menghasilkan model dengan tingkat akurasi minimal 60% dan mencapai *power* hingga 26.
3. Untuk pengklasifikasian 5 kelas gerak, akurasi terbaik yang didapatkan adalah 73.03% dengan rata-rata akurasi 69.90% dari 10 kali percobaan dengan menggunakan Butterwoth *band pass filter* order 4 dengan batas ambang bawah = 29 dan atas = 40 dan lama waktu perekaman data training setiap kelas sebesar 30 detik.

6.2 Saran

Saran yang diberikan untuk pengembangan aplikasi ini adalah:

1. Untuk meningkatkan akurasi, Ada baiknya untuk mencoba hanya memproses data kanal sensor yang berhubungan langsung dengan bagian otak yang yang memproses *mental command*.

2. Selain itu bisa juga mencoba metode *mental command* yang berbeda seperti berpikir untuk menggerakkan anggota badan tertentu.
3. Untuk meningkatkan keleluasaan dalam penggunaan, bisa ditambahkan kontrol kecepatan dengan memanfaatkan sensor *gyro* pada Emotiv EPOC.

DAFTAR PUSTAKA

[1] J. K. Chapin, K. A. Moxon, R. S. Markowitz, and M. A. L. Nicolelis, "Real-time control of a robot arm using simultaneously recorded neurons in the motor cortex," Nature Neurosci., vol. 2, pp. 664–670, 1999.

[2] J. Wessberg, C. R. Stambaugh, J. D. Kralik, P. D. Beck, M. Laubach, J. K. Chapin, J. Kim, S. J. Biggs, M. A. Srinivassan, and M. A. L. Nicolelis, "Real-time prediction of hand trajectory by ensembles of cortical neurons in primates," Nature, vol. 408, pp. 361–365, 2000.

[3] D. M. Taylor, S. I.H. Tillery, and A. B. Schwartz, "Direct cortical control of 3D neuroprosthetic devices," Science, vol. 296, pp. 1829–1832, 2002.

[4] Nugroho AS, Witarto BA dan Handoko D. Support Vector Machine - Teori dan Aplikasinya Dalam Bioinformatika. Kuliah Umum Ilmu Komputer.com. 2003. URL: http://www.ilmukomputer.com/ anto SVM.pdf, diakses tanggal 16 Maret 2008.

[5] Burges JC. A Toturial on Support Vector Machines for Pattern Recognition. *Data Mining and Knowledge Discovery*. 2: 955 974. 1998.

[6] Hsu CW and Lin CJ. A Comparison of Methods for Multi-class Support Vector Machines. IEEE Transactions on Neural network. 13: 415-425. 2002.

[7] J. R. Milln, F. Renkens, J. Mourio, , and W. Gerstner, "Noninvasive brain-actuated control of a mobile robot by human eeg," IEEE Transactions On Biomedical Engineering, vol. 51, no. 6, 2004.

[8] A. Bashashati, M. Fatourechi, R. K. Ward, and G. E. Birch, "A survey of signal processing algorithms in brain-computer interfaces based on electrical brain signals," Journal of Neural Engineering, vol. 4, no. 2, pp. 35–57, 2007.

[9] M. A. L. Nicolelis, “Brain-machine interfaces to restore motor function and probe neural circuits,” Nature Rev. Neurosci., vol. 4, pp. 417–422, 2003.

[10] J. R. Wolpaw, D. J. McFarland, and T. M. Vaughan, “Brain computer interface research at the Wadsworth center,” IEEE Trans. Rehab. Eng., vol. 8, pp. 222–226, June 2000.

[11] G. Pfurtscheller and C. Neuper, “Motor imagery and direct brain-computer communication,” Proc. IEEE, vol. 89, pp. 1123–1134, July 2001.

LAMPIRAN A

Kode Program Mikrokontoler

```
/*****************************************************
This program was produced by the
CodeWizardAVR V2.05.3 Standard
Automatic Program Generator
© Copyright 1998-2011 Pavel Haiduc, HP InfoTech s.r.l.
http://www.hpinfotech.com

Chip type           : ATmega16
Program type        : Application
AVR Core Clock frequency: 12.000000 MHz
Memory model        : Small
External RAM size   : 0
Data Stack size     : 256
*****************************************************/

#include <mega16.h>
#ifndef RXB8
#define RXB8 1
#endif

#ifndef TXB8
#define TXB8 0
#endif

#ifndef UPE
#define UPE 2
#endif

#ifndef DOR
#define DOR 3
#endif

#ifndef FE
#define FE 4
#endif

#ifndef UDRE
#define UDRE 5
#endif

#ifndef RXC
#define RXC 7
```

```
#endif

#define FRAMING_ERROR (1<<FE)
#define PARITY_ERROR (1<<UPE)
#define DATA_OVERRUN (1<<DOR)
#define DATA_REGISTER_EMPTY (1<<UDRE)
#define RX_COMPLETE (1<<RXC)

// USART Receiver buffer
#define RX_BUFFER_SIZE 8
char rx_buffer[RX_BUFFER_SIZE];

#if RX_BUFFER_SIZE <= 256
unsigned char rx_wr_index,rx_rd_index,rx_counter;
#else
unsigned int rx_wr_index,rx_rd_index,rx_counter;
#endif

// This flag is set on USART Receiver buffer overflow
bit rx_buffer_overflow;

// USART Receiver interrupt service routine
interrupt [USART_RXC] void usart_rx_isr(void)
{
char status,data;
status=UCSRA;
data=UDR;
if ((status & (FRAMING_ERROR | PARITY_ERROR | DATA_OVERRUN))==0)
  {
  rx_buffer[rx_wr_index++]=data;
#if RX_BUFFER_SIZE == 256
  // special case for receiver buffer size=256
  if (++rx_counter == 0) rx_buffer_overflow=1;
#else
  if (rx_wr_index == RX_BUFFER_SIZE) rx_wr_index=0;
  if (++rx_counter == RX_BUFFER_SIZE)
    {
    rx_counter=0;
    rx_buffer_overflow=1;
    }
#endif
  }
}

#ifndef _DEBUG_TERMINAL_IO_
// Get a character from the USART Receiver buffer
```

```
#define _ALTERNATE_GETCHAR_
#pragma used+
char getchar(void)
{
char data;
while (rx_counter==0);
data=rx_buffer[rx_rd_index++];
#if RX_BUFFER_SIZE != 256
if (rx_rd_index == RX_BUFFER_SIZE) rx_rd_index=0;
#endif
#asm("cli")
--rx_counter;
#asm("sei")
return data;
}
#pragma used-
#endif

// Standard Input/Output functions
#include <stdio.h>

// Declare your global variables here

void main(void)
{
// Declare your local variables here
char k;

// Input/Output Ports initialization
// Port A initialization
// Func7=In Func6=In Func5=In Func4=In Func3=In Func2=In Func1=In Func0=In
// State7=T State6=T State5=T State4=T State3=T State2=T State1=T State0=T
PORTA=0x00;
DDRA=0x00;

// Port B initialization
// Func7=In Func6=In Func5=In Func4=In Func3=In Func2=In Func1=In Func0=In
// State7=T State6=T State5=T State4=T State3=T State2=T State1=T State0=T
PORTB=0x00;
DDRB=0x00;

// Port C initialization
// Func7=In Func6=In Func5=In Func4=In Func3=In Func2=In Func1=In Func0=In
// State7=T State6=T State5=T State4=T State3=T State2=T State1=T State0=T
PORTC=0x00;
DDRC=0xFF;
```

```
// Port D initialization
// Func7=Out Func6=Out Func5=Out Func4=Out Func3=In Func2=In Func1=Out
Func0=Out
// State7=0 State6=0 State5=0 State4=0 State3=T State2=T State1=0 State0=0
PORTD=0x00;
DDRD=0xFF;

// Timer/Counter 0 initialization
// Clock source: System Clock
// Clock value: Timer 0 Stopped
// Mode: Normal top=0xFF
// OC0 output: Disconnected
TCCR0=0x00;
TCNT0=0x00;
OCR0=0x00;

TCCR1A=0xA1;
TCCR1B=0x0D;
TCNT1H=0x00;
TCNT1L=0x00;
ICR1H=0x00;
ICR1L=0x00;
OCR1AH=0x00;
OCR1AL=0x00;
OCR1BH=0x00;
OCR1BL=0x00;

// Timer/Counter 2 initialization
// Clock source: System Clock
// Clock value: Timer2 Stopped
// Mode: Normal top=0xFF
// OC2 output: Disconnected
ASSR=0x00;
TCCR2=0x00;
TCNT2=0x00;
OCR2=0x00;

// External Interrupt(s) initialization
// INT0: Off
// INT1: Off
// INT2: Off
MCUCR=0x00;
MCUCSR=0x00;

// Timer(s)/Counter(s) Interrupt(s) initialization
```

```
TIMSK=0x00;

// USART initialization
// Communication Parameters: 8 Data, 1 Stop, No Parity
// USART Receiver: On
// USART Transmitter: On
// USART Mode: Asynchronous
// USART Baud Rate: 9600
UCSRA=0x00;
UCSRB=0x98;
UCSRC=0x86;
UBRRH=0x00;
UBRRL=0x4D;

// Analog Comparator initialization
// Analog Comparator: Off
// Analog Comparator Input Capture by Timer/Counter 1: Off
ACSR=0x80;
SFIOR=0x00;

// ADC initialization
// ADC disabled
ADCSRA=0x00;

// SPI initialization
// SPI disabled
SPCR=0x00;

TWCR=0x00;

// Global enable interrupts
#asm("sei")

while (1)
    {
    // Place your code here
    k=getchar(); //Perintah untuk mengambil data
    putchar(k); //Perintah untuk kirim data
    //printf("%c",k);

    if (k=='W')
    {
    PORTD.2=0;
    PORTD.3=1;
    PORTD.6=1;
    PORTD.7=0;
```

```
    OCR1A=255;
    OCR1B=255;
    }

    else if (k=='D')
    {
    PORTD.2=0;
    PORTD.3=1;
    PORTD.6=0;
    PORTD.7=0;
    OCR1A=255;
    OCR1B=255;
    }

    else if (k=='S')
    {
    OCR1A=0;
    OCR1B=0;
    }

    else if (k=='A')
    {
    PORTD.2=0;
    PORTD.3=0;
    PORTD.6=1;
    PORTD.7=0;
    OCR1A=255;
    OCR1B=255;
    }

    else if (k=='X')
    {
    PORTD.2=1;
    PORTD.3=0;
    PORTD.6=0;
    PORTD.7=1;
    OCR1A=255;
    OCR1B=255;
    }
    }
}
```

LAMPIRAN B

Windows1.xaml.cs

//*code behind* dari antarmuka Emotiv Engine Realtime

```
##########################################################
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Navigation;
using System.Windows.Shapes;
using System.Windows.Threading;

using System.IO.Ports;

using Emotiv;
using EmoEngineClientLibrary;
using EmoEngineControlLibrary;
using System.Windows.Automation.Peers;
using System.Reflection;

namespace RawDataTestApp
{
    /// <summary>
    /// Interaction logic for Window1.xaml
    /// </summary>
    public partial class Window1 : Window
    {
        EmoEngineClient _emoEngineClient;
        public AMLearning model;
```

```
        #region variables
        FlowDocument mcFlowDoc = new FlowDocument();
        Paragraph para = new Paragraph();
        #endregion

        public Window1()
        {
            InitializeComponent();

            this._emoEngineClient =
this.FindResource( "emoEngineClient" ) as
EmoEngineClient;
            this._emoEngineClient.ActivePort =
EmoEngineClient.EmoComposerPort; // TBD: move to XAML
            //this._emoEngineClient.ActivePort =
EmoEngineClient.ControlPanelPort;

            this._emoEngineClient.dsp = new
BasicSignalProcessor();
            this._emoEngineClient.ptrUpdateUI =
this.Dispatcher;
            this._emoEngineClient.labelAction =
this.labelAction;
            this._emoEngineClient.btnB = btnBackward;
            this._emoEngineClient.btnF = btnForward;
            this._emoEngineClient.btnN = btnNeutral;
            this._emoEngineClient.btnR = btnRight;
            this._emoEngineClient.btnL = btnLeft;
        }

        private void Connect_Comms(object sender,
RoutedEventArgs e)
        {
            if (Connect_btn.Content == "Connect")
            {
                //Sets up serial port
                this._emoEngineClient.serial.PortName
= Comm_Port_Names.Text;
                this._emoEngineClient.serial.BaudRate
= Convert.ToInt32(Baud_Rates.Text);
```

```
this._emoEngineClient.serial.Handshake =
System.IO.Ports.Handshake.None;
                this._emoEngineClient.serial.Parity =
Parity.None;
                this._emoEngineClient.serial.DataBits
= 8;
                this._emoEngineClient.serial.StopBits
= StopBits.One;

this._emoEngineClient.serial.ReadTimeout = 200;

this._emoEngineClient.serial.WriteTimeout = 50;
                this._emoEngineClient.serial.Open();

                //Sets button State and Creates
function call on data recieved
                Connect_btn.Content = "Disconnect";
                //serial.DataReceived += new
System.IO.Ports.SerialDataReceivedEventHandler(Reciev
e);

            }
            else
            {
                try // just in case serial port is
not open could also be acheved using
if(serial.IsOpen)
                {

this._emoEngineClient.serial.Close();
                    Connect_btn.Content = "Connect";
                }
                catch
                {
                }
            }
        }
```

```
        private void Window_Loaded( object sender,
RoutedEventArgs e )
        {
        }

        private void _startButton_Click( object
sender, RoutedEventArgs e )
        {
            this._emoEngineClient.StartDataPolling();

            this._neuroDataControl.Start();
        }

        private void _stopButton_Click( object
sender, RoutedEventArgs e )
        {
            this._emoEngineClient.StopDataPolling();
        }

        private void _startEmoEngineButton_Click(
object sender, RoutedEventArgs e )
        {


            if (this.model != null)
            {
                this._emoEngineClient.model =
this.model;

this._emoEngineClient.StartEmoEngine();
            }
            else
            {
                MessageBox.Show("Configure data model
first!");
            }
        }
```

```
        private void Window_Closing( object sender,
System.ComponentModel.CancelEventArgs e )
        {
            this._neuroDataControl.Shutdown();
        }

        private static AutoResetEvent s_event = new
AutoResetEvent( false );

        private void btnForward_Click(object sender,
RoutedEventArgs e)
        {
            this._emoEngineClient.SerialCmdSend("W");
        }

        private void btnRight_Click(object sender,
RoutedEventArgs e)
        {
            this._emoEngineClient.SerialCmdSend("D");
        }

        private void btnLeft_Click(object sender,
RoutedEventArgs e)
        {
            this._emoEngineClient.SerialCmdSend("A");
        }

        private void btnBackward_Click(object sender,
RoutedEventArgs e)
        {
            this._emoEngineClient.SerialCmdSend("X");
            //MessageBox.Show("clicked");
        }

        private void btnNeutral_Click(object sender,
RoutedEventArgs e)
        {
            this._emoEngineClient.SerialCmdSend("S");
        }
```

```
        private void _trainData_Click(object sender,
RoutedEventArgs e)
        {

            ClassifyForm cf = new ClassifyForm(this);
            cf.Show();

        }
    }
}
#########################################################
```

ClassifyForm.cs

//kelas *form* Configuration Data Model

```
#########################################################
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using System.Threading;
using EmoEngineClientLibrary;
using Accord.Statistics.Analysis;
using RawDataTestApp.Algorithms;
using NLog;

namespace RawDataTestApp
{
    public partial class ClassifyForm : Form
    {
        AMLearning model;

        //IFeatureGenerator fg;

        Window1 home;

        List<AMLearning> models;
```

```
        BackgroundWorker AsyncWorkerLoadModels;

        BackgroundWorker AsyncWorkerProcess;

        Logger logger =
LogManager.GetCurrentClassLogger();

        ModelStorage ms;

        public ClassifyForm(Window1 w)
        {
            InitializeComponent();

            listBoxModels.SelectedIndex = -1;
            home = w;

            AsyncWorkerLoadModels = new
BackgroundWorker();

AsyncWorkerLoadModels.WorkerReportsProgress = true;

AsyncWorkerLoadModels.WorkerSupportsCancellation =
true;
            AsyncWorkerLoadModels.RunWorkerCompleted
+= new
RunWorkerCompletedEventHandler(AsyncWorkerLoadModels_
RunWorkerCompleted);
            AsyncWorkerLoadModels.DoWork += new
DoWorkEventHandler(AsyncWorkerLoadModels_DoWork);

            toolStripStatusLabel1.Text = "Loading
models. Please wait...";
            ms = new ModelStorage();
            AsyncWorkerLoadModels.RunWorkerAsync();

            AsyncWorkerProcess = new
BackgroundWorker();
            AsyncWorkerProcess.WorkerReportsProgress
= true;
```

```
AsyncWorkerProcess.WorkerSupportsCancellation = true;
            AsyncWorkerProcess.ProgressChanged += new
ProgressChangedEventHandler(AsyncWorkerProcess_Progre
ssChanged);
            AsyncWorkerProcess.DoWork += new
DoWorkEventHandler(AsyncWorkerProcess_DoWork);
            AsyncWorkerProcess.RunWorkerCompleted +=
new
RunWorkerCompletedEventHandler(AsyncWorkerProcess_Run
WorkerCompleted);

                    this.FormClosing += new
FormClosingEventHandler(ClassifyForm_FormClosing);
            //listBoxResult.Items.Insert(0, "You are
required to wait until you see: \"Initialization took
xxxxx ms\" in OpenVibe.");
        }

        public void RefreshData()
        {
            listBoxModels.Items.Clear();
            AsyncWorkerLoadModels.RunWorkerAsync();
        }

        void
AsyncWorkerProcess_RunWorkerCompleted(object sender,
RunWorkerCompletedEventArgs e)
        {
            if (e.Error != null)
            {

System.Windows.Forms.MessageBox.Show("Error:" +
e.Error.Message+" "+e.Error.StackTrace);
                //listBoxResult.Items.Insert(0,
"Classification failed.");
                logger.Error(e.Error);
            }
            else
            {
```

```
                //listBoxResult.Items.Insert(0,
"Classification process is done.");
            }
            buttonStartProcessing.Enabled = true;
            buttonStartProcessing.Text = "Process";
        }

        void
AsyncWorkerProcess_ProgressChanged(object sender,
ProgressChangedEventArgs e)
        {

//listBoxResult.Items.Insert(0,(string)e.UserState);
        }

        void AsyncWorkerProcess_DoWork(object sender,
DoWorkEventArgs e)
        {
            while
(!AsyncWorkerProcess.CancellationPending)
            {
                //fg.Update();
            }

            if
(AsyncWorkerProcess.CancellationPending) e.Cancel =
true;
        }

        void
AsyncWorkerLoadModels_RunWorkerCompleted(object
sender, RunWorkerCompletedEventArgs e)
        {
            if (e.Error != null)
            {

       System.Windows.Forms.MessageBox.Show("Could
not load available models! Database error:" +
e.Error.Message);
```

```
                toolStripStatusLabel1.Text = "No
models loaded.";
                logger.Error(e.Error);
                return;
            }
            else
            {
                if (models != null && models.Count >
0)
                {
                    string[] names = (from m in
models
                                      where m != null && m.Name
!= null
                                      select m.Name).ToArray();


listBoxModels.Items.AddRange(names);

                    toolStripStatusLabel1.Text =
"Models loaded: " + names.Length;
                }
                else toolStripStatusLabel1.Text = "No
models loaded.";
            }
        }

        void AsyncWorkerLoadModels_DoWork(object
sender, DoWorkEventArgs e)
        {
            models = ms.LoadModels();
        }

        /// <summary>
        /// Start processing signal and
classification
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
```

```
        private void
buttonStartProcessing_Click(object sender, EventArgs
e)
        {

            textBoxInfo.Clear();
            textBoxInfo.Text = "";
            home.model = this.model;
            textBoxInfo.Text += "Name\t\t: " +
this.model.Name;
            textBoxInfo.Text += "\r\nAlgorithms\t: "
+ this.model.Algo;

            if(this.model.Filter==1)
                textBoxInfo.Text += "\r\nFilter\t\t:
ButterworthBandPass(min=29, max=40 , order 4)";
            else
                textBoxInfo.Text += "\r\nFilter\t\t:
none";

            string listClass = "\r\nTrained class\t:
" + listBoxClasses.Items.Count;
            for(int
i=0;i<listBoxClasses.Items.Count;i++)
            {
                listClass += "\r\n\t-" +
listBoxClasses.Items[i];
            }
            textBoxInfo.Text += listClass;
        }

        private void
listBoxModels_SelectedIndexChanged(object sender,
EventArgs e)
        {
            if (listBoxModels.SelectedIndex != -1)
            {
                model =
models[listBoxModels.SelectedIndex];
```

```
                listBoxClasses.Items.Clear();

                foreach (var item in
model.ActionList)
                {

listBoxClasses.Items.Add(item.Key);
                }
            }
        }

        private void buttonClose_Click(object sender,
EventArgs e)
        {
            home.model = this.model;
            Clear();
            this.Close();
        }

              void ClassifyForm_FormClosing(object
sender, FormClosingEventArgs e)
              {
                    Clear();
              }

              void Clear()
              {
                    if
(!AsyncWorkerLoadModels.CancellationPending)
                    {

       AsyncWorkerLoadModels.CancelAsync();
                    }

                    if
(!AsyncWorkerProcess.CancellationPending)
                    {

       AsyncWorkerProcess.CancelAsync();
                    }
```

```
                    //this.fg=null;
                    //AsyncWorkerProcess = null;
                    //AsyncWorkerLoadModels = null;
            }

        private void button1_Click(object sender,
EventArgs e)
        {

        }

        void rd_ReocordSelected(EEGRecord record)
        {
            foreach(double[] vector in
record.FeatureVectorsOutputInput)
            {
                double[] input = new
double[vector.Length-1];

                Array.Copy(vector, 1, input, 0,
vector.Length - 1);

                int result = model.Classify(input);
                //if (result == vector[0])
                    //listBoxResult.Items.Insert(0,
"OK");
                //else
                    //listBoxResult.Items.Insert(0,
"wrong");
            }

        }

        private void buttonTrain_Click(object sender,
EventArgs e)
        {
            TrainForm tf = new TrainForm(this);
            tf.Show();
        }
```

```
        private void ClassifyForm_Load(object sender,
EventArgs e)
        {

        }
    }
}
##########################################################
```

TrainForm.cs

//kelas *form* Create Model

```
##########################################################
using System;

using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using System.Threading;
using System.IO;

using Accord.Statistics.Analysis;
using AForge.Neuro;
using AForge.Neuro.Learning;
using Eloquera.Client;
using RawDataTestApp.Algorithms;
using NLog;
using EmoEngineClientLibrary;
namespace RawDataTestApp
{
    public partial class TrainForm : Form
    {
        #region declarations
        IFeatureGenerator fg;
        private IRawDataReader dataReader;
```

```
        AMLearning model;
        IDigitalSignalProcessor dsp;
        EEGRecord currentRecord = new EEGRecord();

        private BackgroundWorker
AsyncWorkerCalculate;

        private BackgroundWorker AsyncWorkerRecord;

        private BackgroundWorker
AsyncWorkerSaveModel;

        private int LastRecodredFeatureVectorsCount;

        DateTime startCalculateModel;

        ModelStorage ms;

        private static Logger logger =
LogManager.GetCurrentClassLogger();
        #endregion

        /// <summary>
        /// Increases after each recording. It is
different from comboBoxSelectedClass.SelectedIndex
        /// </summary>
        ///
        ClassifyForm classifyForm;

        System.IO.StreamReader file;
        int SelectedClassNumeric = 0;

        int recordTime;
        DateTime startRecord;
        System.Timers.Timer recordTimer;

        public TrainForm(ClassifyForm cf)
        {
            InitializeComponent();
```

```
            this.classifyForm = cf;

            ResetCheckBoxAction();

            comboBoxSelectedClass.SelectedIndex = 0;
            comboBoxRecordTime.SelectedIndex = 0;

       comboBoxRecordTime.SelectedIndexChanged += new
EventHandler(comboBoxRecordTime_SelectedIndexChanged)
;

            recordTime =
Convert.ToInt32(comboBoxRecordTime.Text);
            recordTimer = new System.Timers.Timer();
            recordTimer.Enabled = true;
            recordTimer.Interval = 1000;


            AsyncWorkerCalculate = new
BackgroundWorker();

AsyncWorkerCalculate.WorkerReportsProgress = true;

AsyncWorkerCalculate.WorkerSupportsCancellation =
true;
            AsyncWorkerCalculate.ProgressChanged +=
new
ProgressChangedEventHandler(AsyncWorkerCalculate_Prog
ressChanged);
            AsyncWorkerCalculate.RunWorkerCompleted
+= new
RunWorkerCompletedEventHandler(AsyncWorkerCalculate_R
unWorkerCompleted);
            AsyncWorkerCalculate.DoWork += new
DoWorkEventHandler(AsyncWorkerCalculate_DoWork);

            AsyncWorkerRecord = new
BackgroundWorker();
```

```
            AsyncWorkerRecord.WorkerReportsProgress =
true;

AsyncWorkerRecord.WorkerSupportsCancellation = true;
            AsyncWorkerRecord.ProgressChanged += new
ProgressChangedEventHandler(AsyncWorkerRecord_Progres
sChanged);
            AsyncWorkerRecord.RunWorkerCompleted +=
new
RunWorkerCompletedEventHandler(AsyncWorkerRecord_RunW
orkerCompleted);
            AsyncWorkerRecord.DoWork += new
DoWorkEventHandler(AsyncWorkerRecord_DoWork);

            AsyncWorkerSaveModel = new
BackgroundWorker();

AsyncWorkerSaveModel.WorkerReportsProgress = true;

AsyncWorkerSaveModel.WorkerSupportsCancellation =
true;
            AsyncWorkerSaveModel.RunWorkerCompleted
+= new
RunWorkerCompletedEventHandler(AsyncWorkerSaveModel_R
unWorkerCompleted);
            AsyncWorkerSaveModel.DoWork += new
DoWorkEventHandler(AsyncWorkerSaveModel_DoWork);

            ms = new ModelStorage();
            cbMethods.SelectedIndex = 0;

        }

        void ResetCheckBoxAction()
        {
            checkBoxNeutral.Checked = false;
            checkBoxTurnRight.Checked = false;
            checkBoxTurnLeft.Checked = false;
            checkBoxForward.Checked = false;
            checkBoxBackward.Checked = false;
```

```
        }

            void
comboBoxRecordTime_SelectedIndexChanged(object
sender, EventArgs e)
            {
                    recordTime =
Convert.ToInt32(comboBoxRecordTime.Text);
            }

        void
AsyncWorkerCalculate_ProgressChanged(object sender,
ProgressChangedEventArgs e)
        {
            progressBarModelCalculation.Value =
e.ProgressPercentage;
        }

        void AsyncWorkerSaveModel_DoWork(object
sender, DoWorkEventArgs e)
        {
            ms.SaveModel(model);
        }

        void
AsyncWorkerSaveModel_RunWorkerCompleted(object
sender, RunWorkerCompletedEventArgs e)
        {
            if (e.Error != null)
            {
                MessageBox.Show("Error: " +
e.Error.Message + " " + e.Error.StackTrace);
                logger.Error(e.Error);
                listBoxLogger.Items.Insert(0, "Error
saving model!");
            }
            else
            {
                listBoxLogger.Items.Insert(0, "Model
'" + textBoxModelName.Text + "' saved.");
```

```
            }
            buttonSaveModel.Enabled = true;

        }

        void recordTimer_Tick(object sender,
EventArgs e)
        {
            throw new NotImplementedException();
        }

        void AsyncWorkerRecord_ProgressChanged(object
sender, ProgressChangedEventArgs e)
        {
            for (int i = 0; i < recordTime; i++)
            {
                int percentCompleted = (i *
100)/recordTime;
                //progressBarRecord.Value =
percentCompleted;
            }
        }

        void SetCheckBoxAction(int index)
        {
            if(index==0)
            {
                checkBoxNeutral.Checked = true;
            }
            else if(index==1)
            {
                checkBoxTurnRight.Checked = true;
            }
            else if (index == 2)
            {
                checkBoxTurnLeft.Checked = true;
            }
            else if (index == 3)
            {
                checkBoxForward.Checked = true;
```

```
            }
            else if (index == 4)
            {
                checkBoxBackward.Checked = true;
            }
        }

        void
AsyncWorkerRecord_RunWorkerCompleted(object sender,
RunWorkerCompletedEventArgs e)
        {
            if (e.Error != null)
            {
                MessageBox.Show("Error: " +
e.Error.Message + " " + e.Error.StackTrace);
                logger.Error(e.Error);
                listBoxLogger.Items.Insert(0, "Error
recording data!");
            }
            else
            {
                listBoxLogger.Items.Insert(0,
"Recording completed. " +
LastRecodredFeatureVectorsCount.ToString() + "
additional vectors acquired.");

SetCheckBoxAction(comboBoxSelectedClass.SelectedIndex
);
            }
            buttonRecordAction.Enabled = true;
        }

        void AsyncWorkerRecord_DoWork(object sender,
DoWorkEventArgs e)
        {
            if (checkBoxEnableBasicDSP.Checked)
                this.dsp = new
BasicSignalProcessor();
            else
                this.dsp = null;
```

```
            LastRecodredFeatureVectorsCount = 0;

            startRecord = DateTime.Now;
            recordTimer.Start();

            double[] result = new double[15];
            for (int k = 0; k < this.recordTime *
128; k++)
            {
                string[] columns =
this.file.ReadLine().Split(',');
                double[] dataForDSP = new double[14];

                result[0] = SelectedClassNumeric;
                for (int i = 1; i < 15; i++)
                {
                    result[i] =
double.Parse(columns[i + 2],
System.Globalization.CultureInfo.InvariantCulture);
                    dataForDSP[i - 1] = result[i];
                }

                if (dsp != null)
                {
                    dsp.DoWork(ref dataForDSP);
                    for (int i = 1; i < 15; i++)
                    {
                        result[i] = dataForDSP[i -
1];
                    }
                }


currentRecord.FeatureVectorsOutputInput.Add(result);
                LastRecodredFeatureVectorsCount++;

            }
        }
```

```
        void recordTimer_Elapsed(object sender,
System.Timers.ElapsedEventArgs e)
        {
            TimeSpan elapsedTime = e.SignalTime -
startRecord;

            if (elapsedTime.TotalMilliseconds >=
(recordTime * 1000))
            {
                recordTimer.Stop();
                AsyncWorkerRecord.CancelAsync();
            }
            else
            {
                int percentCompleted =
Convert.ToInt32((elapsedTime.TotalMilliseconds /
(recordTime * 1000)) * 100);
                //ReportProgress(percentCompleted);
            }
        }

        void
AsyncWorkerCalculate_RunWorkerCompleted(object
sender, RunWorkerCompletedEventArgs e)
        {
            TimeSpan elapsed = DateTime.Now -
startCalculateModel;
            string elapsedTime = elapsed.Hours + "
hour(s), " + elapsed.Minutes + " minute(s), " +
elapsed.Seconds + " second(s), " +
elapsed.Milliseconds + " millisecond(s).";

            if (e.Error != null)
            {
                if (e.Error is
InvalidRecordException)

MessageBox.Show(e.Error.Message,"Error");
                else MessageBox.Show(e.Error.Message
+ " " + e.Error.StackTrace, "Error");
```

```
                logger.Error(e.Error);
                listBoxLogger.Items.Insert(0,
"Calculating model has been aborted! Time elapsed: "
+ elapsedTime);
            }
            else
            {
                listBoxLogger.Items.Insert(0,
"Calculating model has completed successfully. Time
elapsed: " + elapsedTime);
            }
            buttonCalculate.Enabled = true;
        }

        void AsyncWorkerCalculate_DoWork(object
sender, DoWorkEventArgs e)
        {
            model.Train(currentRecord);
        }

        void model_Progress(int progress)
        {

AsyncWorkerCalculate.ReportProgress(progress);
        }

        private void buttonRecordAction_Click(object
sender, EventArgs e)
        {

            if (file == null) return;

            if (AsyncWorkerRecord.IsBusy)
            {
                buttonRecordAction.Enabled = false;

                AsyncWorkerRecord.CancelAsync();
            }
            else
            {
```

```
                string ClassName =
comboBoxSelectedClass.Items[comboBoxSelectedClass.Sel
ectedIndex].ToString();

                if
(!currentRecord.actions.Keys.Contains(ClassName))
                {
                    if (currentRecord.actions.Count
== 0) SelectedClassNumeric = 1;
                    else SelectedClassNumeric =
currentRecord.actions.Values.Max() + 1; //choose a
new class (not yet used)


currentRecord.actions.Add(ClassName,
SelectedClassNumeric);
                }
                else
                {   //user already recorded for this
class
                    SelectedClassNumeric =
currentRecord.actions[ClassName];
                }

                listBoxLogger.Items.Insert(0,
"Recoding data for action \"" +
comboBoxSelectedClass.Text + "\" (class " +
SelectedClassNumeric + ").");

                AsyncWorkerRecord.RunWorkerAsync();


            }

        }

        private void buttonCalculate_Click(object
sender, EventArgs e)
        {
```

```
            {

listBoxLogger.Items.Insert(0,"Creating machine
learning model to be used for classification...");
                if
(currentRecord.FeatureVectorsOutputInput.Count == 0)
{ MessageBox.Show("First you need to record/load some
data for specific action!"); return; }


                switch (cbMethods.SelectedIndex)
                {



                    case 0: model = new
LdaSVM("Support vector machines"); break;
                    case 1: model = new
LdaMLP("Multi-layer perceptron with backpropagation
as learning method"); break;
                    case 2: model = new
LdaRBF("Radial basis function with resilient
propagation as learning method"); break;
                    //case 3: model = new
OctaveMulticlassLogisticRegression("omlr"); break;
                }
                model.ActionList =
currentRecord.actions;
                model.Progress += new
ChangedValuesEventHandler(model_Progress);


AsyncWorkerCalculate.RunWorkerAsync();

            }
        }

        private void buttonSaveModel_Click(object
sender, EventArgs e)
```

```
        {
            if (AsyncWorkerCalculate.IsBusy)
            {
                buttonSaveModel.Enabled = false;

                AsyncWorkerSaveModel.CancelAsync();
            }
            else
            {
                listBoxLogger.Items.Insert(0, "Saving
model ... please wait!");

                if (textBoxModelName.Text.Length ==
0)
                { MessageBox.Show("Please enter model
name!"); return; }

                if (model == null)
                { MessageBox.Show("You need to
calculate model first!"); return; }

                buttonSaveModel.Enabled = false;

                if (checkBoxEnableBasicDSP.Checked)
                    model.Filter = 1;
                model.Name = textBoxModelName.Text;
                model.ActionList =
currentRecord.actions;


AsyncWorkerSaveModel.RunWorkerAsync();
            }
        }

        private void buttonCloseForm_Click(object
sender, EventArgs e)
        {
            this.classifyForm.RefreshData();
            this.Close();
        }
```

```
        private void
buttonManageRecordings_Click(object sender, EventArgs
e)
        {
        }

        void rd_ReocordSelected(EEGRecord
selectedRecord)
        {
            currentRecord = new
EEGRecord(selectedRecord);//create a copy
            listBoxLogger.Items.Insert(0, "Pre-
recorded data '" + selectedRecord.Name + "' has been
loaded containing:
"+currentRecord.FeatureVectorsOutputInput.Count+"
feature vectors. You can now record additional data
or start 'Computing'.");
        }

        private void buttonClearRecord_Click(object
sender, EventArgs e)
        {
            currentRecord = new EEGRecord();
            ResetCheckBoxAction();
                    listBoxLogger.Items.Insert(0,
"Recorded data cleared. Now you can record or load
data.");

        }

        private void TrainForm_Load(object sender,
EventArgs e)
        {

        }

        private void
cbMethods_SelectedIndexChanged(object sender,
EventArgs e)
```

```
        {

        }

        private void buttonBrowse_Click(object
sender, EventArgs e)
        {
            OpenFileDialog fo = new OpenFileDialog();
            DialogResult result = fo.ShowDialog();
            if (result == DialogResult.OK)
            {
                try
                {
                    textBoxEmotivFile.Text =
fo.FileName;
                    this.file = new
System.IO.StreamReader(textBoxEmotivFile.Text);
                    file.ReadLine();//skip one line


                }
                catch (Exception ex)
                {
                    MessageBox.Show("Error:" +
ex.Message);
                }
            }

        }

        private void btnTest_Click(object sender,
EventArgs e)
        {
            foreach (double[] vector in
currentRecord.FeatureVectorsOutputInput)
            {
                double[] input = new
double[vector.Length - 1];
                Array.Copy(vector, 1, input, 0,
vector.Length - 1);
```

```
                int result = model.Classify(input);
                if (result == vector[0])
                    listBoxLogger.Items.Insert(0,
result + " " + vector[0] + " OK");
                else
                    listBoxLogger.Items.Insert(0,
result + " " + vector[0] + " Wrong");
            }
        }

        private void btnVisualizeData_Click(object
sender, EventArgs e)
        {
            if (checkBoxEnableBasicDSP.Checked)
            {
                this.dsp = new
BasicSignalProcessor();
                dataReader = new
EmotivFileSystemDataReader(textBoxEmotivFile.Text,
dsp);
            }
            else
                dataReader = new
EmotivFileSystemDataReader(textBoxEmotivFile.Text);

            WPF.OutputWindow ow = new
WPF.OutputWindow(dataReader); ow.Show();
        }

        private void groupBox1_Enter(object sender,
EventArgs e)
        {

        }


    }
```

[Halaman ini sengaja dikosongkan]

LAMPIRAN C

Tabel C.1. Rekap Hasil Keluaran Program Skenario Uji Coba 1

Percobaan	Maju	Mundur	Kanan	Kiri	Netral	Akurasi	Power Maksimal
1	38.33	39.93	47.36	42.21	73.32	48.23	16
2	58.15	59.97	61.25	54.98	72.5	61.37	15
3	56.52	58.88	49.52	58.97	71.71	59.12	18
4	47.53	50.55	38.43	47.21	76.33	52.01	12
5	61.77	59.13	55.38	62.18	75.94	62.88	12
6	41.32	42.21	45.19	44.61	53.52	45.37	18
7	46.51	46.29	37.61	45.09	80.45	51.19	10
8	51.08	59.29	58.39	49.13	81.76	59.93	11
9	40.37	46.07	46.7	38.54	74.42	49.22	10
10	38.23	41.51	43.93	43.47	59.01	45.23	12
Mean	47.981	50.383	48.376	48.639	71.896	53.455	13.4
Std	8.62136	8.233603	7.923055	7.684387	8.938107	6.757446	3.098386677

Tabel C.2. Rekap Hasil Keluaran Program Skenario Uji Coba 2

Percobaan	Maju	Mundur	Kanan	Kiri	Netral	Akurasi	Power Maksimal
1	64.63	66.22	62.7	63.4	87.35	68.86	21
2	68.51	61.07	59.85	68.49	85.43	68.67	19
3	68.08	68.26	68.44	68.86	73.36	69.4	18
4	65.57	62.32	60.15	60.65	96.16	68.97	19
5	64.52	67.55	56.74	67.44	89.75	69.2	18
6	71.05	72.33	62.14	72.27	87.36	73.03	25
7	66.57	66.16	71.44	64.77	90.46	71.88	20
8	68.82	67.22	63.52	69.03	77.01	69.12	21
9	68.07	69.92	68.3	69	76.11	70.28	26
10	69.13	65.83	66.53	68.77	79.19	69.89	21
Mean	67.495	66.688	63.981	67.268	84.218	69.93	20.8
Std	2.118439	3.293943	4.595736	3.373708	7.410198	1.439931	2.740640639

Tabel C.3. Rekap Hasil Keluaran Program Skenario Uji Coba 3

Percobaan	Maju	Mundur	Kanan	Kiri	Netral	Akurasi	Power Maksimal
1	61.11	58.96	61.09	63.56	74.08	63.76	18
2	54.32	57.93	60.53	61.05	73.02	61.37	17
3	63.79	63.35	64.01	60.21	81.59	66.59	17
4	64.38	66.49	69.29	62.69	83.8	69.33	18
5	65.39	69.99	68.25	64.59	83.23	70.29	21
6	70.99	64.59	66.81	71.21	85.75	71.87	20
7	69.02	60.5	61.31	64.78	90.54	69.23	21
8	70.83	66.78	70.02	65.8	81.47	70.98	24
9	64.53	65.06	67.45	58.89	90.72	69.33	21
10	67.56	54.75	69.5	68.95	87.24	69.6	21

Tabel C.4. Rekap Hasil Keluaran Program Skenario Uji Coba 4

Percobaan	Maju	Mundur	Kanan	Kiri	Netral	Akurasi	Power Maksimal
1	44.69	44.5	44.65	45.17	48.74	45.55	10
2	55.94	47.63	55.87	55.76	79.35	58.91	15
3	63.98	66.87	63.63	65.42	82.25	68.43	17
4	64.14	66.97	66.7	61.42	77.87	67.42	18
5	59.67	66.13	66.68	57.57	86.55	67.32	21
6	55.59	67.95	65.05	60.03	99.83	69.69	19
7	61.81	56.1	63.28	63	74.31	63.7	16
8	65.12	64.01	64.49	60.22	72.01	65.17	19
9	70.14	69.86	64.17	66.2	81.03	70.28	21
10	67.07	62.43	60.87	66.25	86.13	68.55	19

Tabel C.5.1. Rekap Hasil Keluaran Program Skenario Uji Coba 5 variasi 1.

Percobaan	Netral	Maju	Akurasi	Power Maksimal
1	92.23	86.01	89.12	33
2	89.77	87.81	88.79	38
3	86.35	86.88	86.615	38
4	88.95	89.23	89.09	37
5	88.05	86.75	87.4	35
6	86.78	95.04	90.91	34
7	95.33	83.54	89.435	38
8	91.64	94.41	93.025	35
9	88.79	88.81	88.8	34
10	86.9	95.75	91.325	32
mean	89.479	89.423	89.451	35.4
std	2.849559	4.20762	1.878664	2.221110833

Tabel C.5.2. Rekap Hasil Keluaran Program Skenario Uji Coba 5 variasi 2.

Percobaan	Netral	Maju	Mundur	Akurasi	Power Maksimal
1	81.83	84.84	83.18	83.28333333	24
2	86.32	84.43	75.61	82.12	31
3	81.57	83.9	78.16	81.21	28
4	85.63	78.97	84.23	82.94333333	30
5	77.44	84.25	89.91	83.86666667	25
6	88.62	86.53	83.05	86.06666667	31
7	75.31	79.41	82.05	78.92333333	30
8	84	90.14	81.11	85.08333333	25
9	80.71	75.19	89.61	81.83666667	24
10	80.43	80.37	78.24	79.68	24
mean	82.186	82.803	82.515	82.50133333	27.2
std	4.071721	4.319174	4.664352	2.237709477	3.084008935

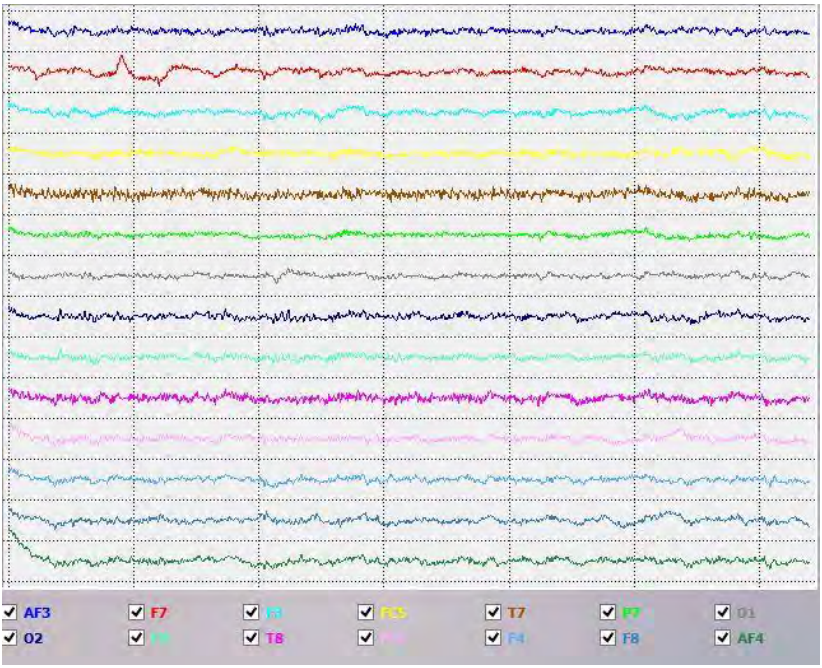
Tabel C.5.3. Rekap Hasil Keluaran Program Skenario Uji Coba 5 variasi 3.

Percobaan	Netral	Maju	Kanan	Kiri	Akurasi	Power Maksimal
1	92.23	70.1	71.6	66.83	75.19	23
2	89.77	65.88	66.71	72.01	73.5925	25
3	86.35	73.2	75.44	70.85	76.46	26
4	88.95	78.28	75.03	69	77.815	21
5	88.05	72.93	76.87	66.81	76.165	25
6	86.78	79.13	79.17	66.44	77.88	20
7	95.33	72.64	66.02	74.23	77.055	24
8	91.64	68.55	67.66	80.19	77.01	23
9	88.79	72	70.48	72.04	75.8275	23
10	86.9	69.24	80.49	67.47	76.025	29
mean	89.479	72.195	72.947	70.587	76.302	23.9
std	2.849559	4.125719	5.210063	4.314149974	1.2811051	2.558211181

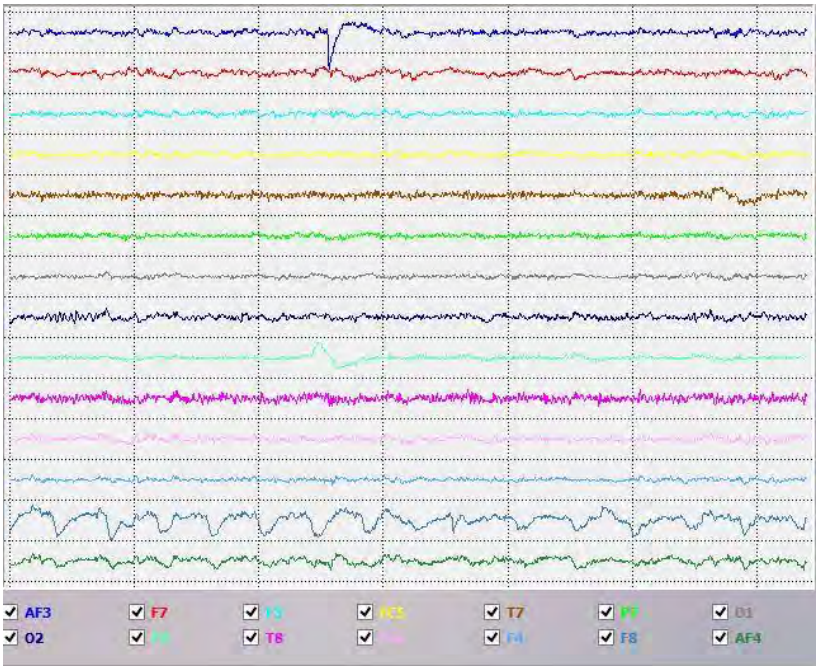
Tabel C.5.4. Rekap Hasil Keluaran Program Skenario Uji Coba 5 variasi 4.

Percobaan	Netral	Maju	Mundur	Kanan	Kiri	Akurasi	Power Maksimal
1	83.25	57.11	56.55	62.83	65.84	65.116	19
2	82.94	58.2	75.01	56.12	58.75	66.204	18
3	78.76	55.74	62.36	62.92	74.93	66.942	19
4	84.5	56.57	66.71	70.64	63.31	68.346	23
5	75.54	67.4	60.66	69.73	72.54	69.174	20
6	73.2	58.87	59.75	74.11	75.84	68.354	18
7	73.64	75.89	63.95	70.81	72.37	71.332	19
8	81.97	64.01	60.7	70.42	58.63	67.146	25
9	80.18	56.25	71.92	64.76	70.64	68.75	22
10	72.86	56.11	60.83	69.42	60.03	63.85	22
mean	78.684	60.615	63.844	67.176	67.288	67.5214	20.5
std	4.535652	6.595672	5.767715	5.3794	6.78758	2.150237827	2.368778401

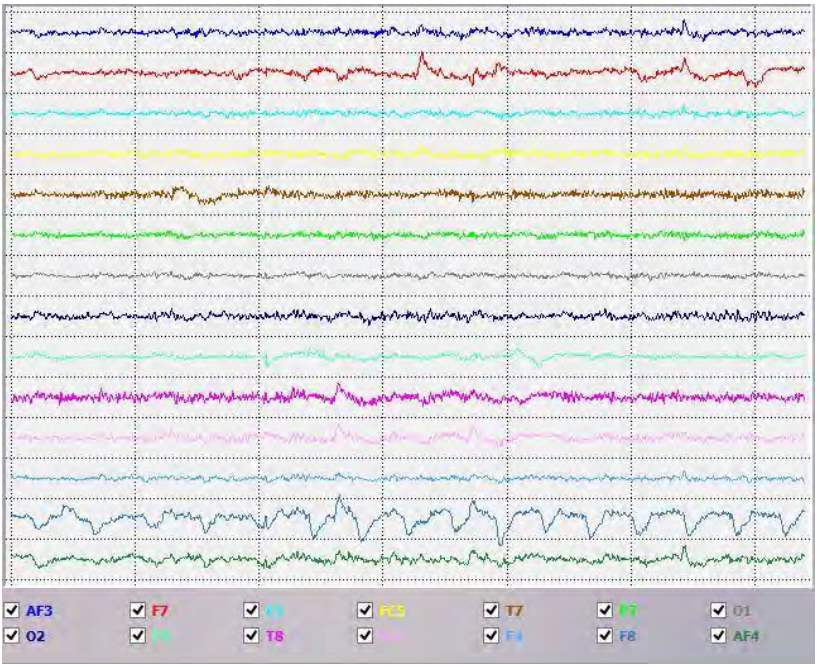
LAMPIRAN D



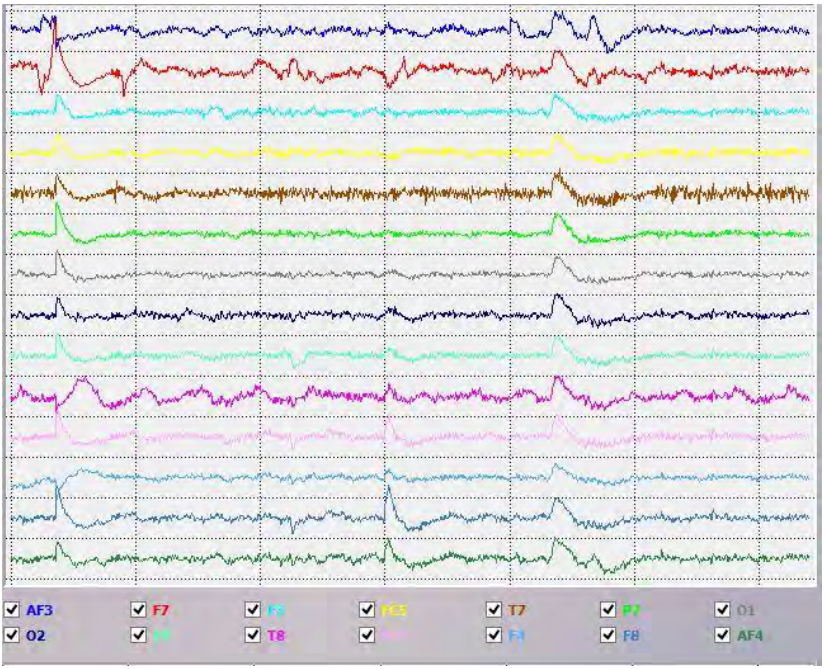
Gambar D.1. Visualisasi gelombang EEG kelas netral



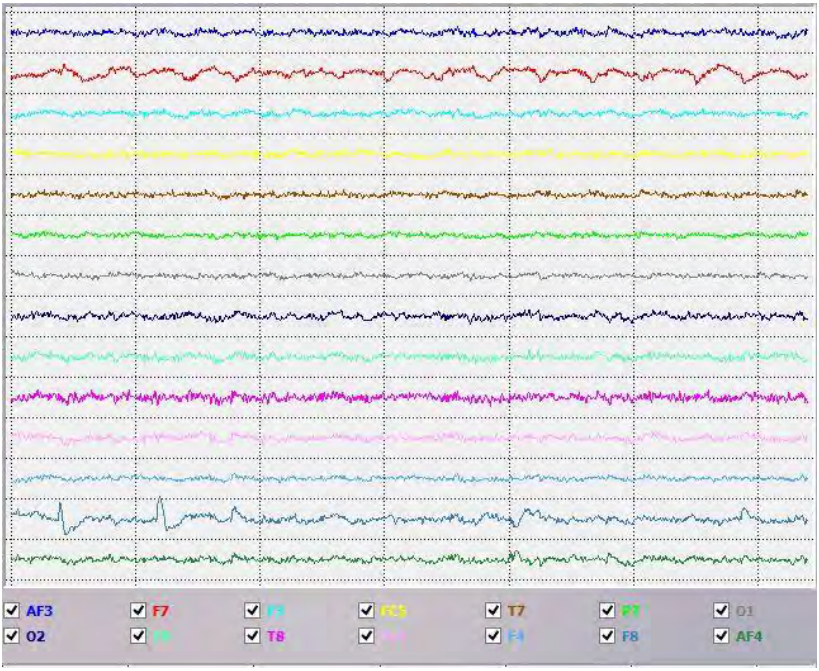
Gambar D.2. Visualisasi gelombang EEG kelas maju.



Gambar D.3. Visualisasi gelombang EEG kelas mundur.



Gambar D.5. Visualisasi gelombang EEG kelas kanan.



Gambar D.6. Visualisasi gelombang EEG kelas kiri.

[Halaman ini sengaja dikosongkan]

BIODATA PENULIS

Sandy Akbar Dewangga, lahir di Nganjuk, pada tanggal 29 Desember 1992. Penulis menempuh pendidikan mulai dari SDN Sukorejo 2 Rejoso-Nganjuk (1999-2001), SDN Rongtengah 1 Sampang (2001-2005), SMP Negeri 1 Sampang, (2005-2008), SMA Negeri 1 Sampang (2008-2011) dan S1 Teknik Informatika ITS (2011-2015).

Selama masa kuliah, penulis aktif dalam organisasi Himpunan Mahasiswa Teknik Computer (HMTC). Diantaranya adalah menjadi staff departemen kesejahteraan mahasiswa himpunan mahasiswa teknik computer ITS 2012-2013. Penulis juga aktif dalam kegiatan kepanitiaan Schematics. Diantaranya penulis pernah menjadi anggota tim soal NLC (*National Logic Competition*) Schematics 2013. Selain itu penulis juga aktif menjadi Innovator di Nokia Innovation Center dan Administrator di Microsoft Mobility Lab.

Selama kuliah di teknik informatika ITS, penulis mengambil bidang minat Komputasi Cerdas Visual (KCV). Penulis pernah menjadi asisten dosen mata kuliah pemrograman perangkat bergerak. Komunikasi dengan penulis dapat melalui email: **sssandy.ad@gmail.com**.